Computers and Video: Coincidental in Architectural Design

by

Janet Elizabeth Gardner

Bachelor of Architecture
Ball State University
Muncie, Indiana
1985

SUBMITTED TO THE DEPARTMENT OF ARCHITECTURE
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE

MASTER OF SCIENCE IN ARCHITECTURE STUDIES
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 1987

Signature of Author _____
Janet Elizabeth Gardner
Department of Architecture
May 8, 1987

Certified By _____
James Anderson
Lecturer, Department of Architecture
Thesis Supervisor

Accepted by _____
Julian Beinart
Chairman
Departmental Committee for Graduate Students

Computers and Video: Coincidental in Architectural Design

By
Janet Elizabeth Gardner

Submitted to the Department of Architecture on May 8, 1987
in partial fulfillment of the requirements for the Degree of
Master of Science in Architecture Studies

ABSTRACT

Architects express abstract concepts in physical concrete realities. The tools of the trade act as a medium through which to express these concepts; pencil, paper, models. Yet these tools do not allow for any intelligent discourse. While translating an abstract idea into a physical object, a designer must keep track of complex associations between various elements within a design. As an idea becomes further developed, the associations between elements become more difficult to manage. On occasion matrices are employed to maintain a consistent relational system, yet complexity often forces design concepts to get distorted or lost. This thesis provides one solution; a method for to the management of this barrage of associative information, while still providing a stimulating design environment.

With the proposed tool, a designer can use numerous media simultaneously; for our purposes digital video and an object-oriented graphics editor. The video environment acts as an analysis tool for exploring existing architypes, in search ofinherent themes. The graphics editor is a tool for synthesizing objects, creating a whole from a series of parts discovered in the video environment.

The key to this tool is it method for handling objects. Objects have the ability to retain knowledge, therefore acting as an intelligent adjunct in the design process. The architect designs with objects. Objects "know" what their behavior is, and how to perform certain operations. Other attributes and relationships can also be attached to objects, or classes of objects.

Compilation of both environments is possible, due to a central knowledge base that maintains all the information from all objects. This single knowledge base allows objects in one environment to be shared by the other, creating an interactive environment between two independent media, computers and video, therefore eliminating the bold line that typically separates the two.

Thesis Supervisor: James Anderson
Title: Lecturer, Department of Architecture

ACKNOWLEDGMENTS

My sincere thanks to the following people:

To my family, for their continued support in all my endeavors, direct and indirect.

To James Anderson, my thesis advisor, for his pointed questions, and his critical help at important times both in writing and in programming.

To Stephen Ervin for his words of encouragement, who's hours of patience are greatly appreciated.

To Professor John Habraken, for exposing me to a design methodology that was well adapted to the exploration of this thesis.

To all those in M.I.T.'s Computer Resource Laboratory past and present, for the introduction to many new and exciting concepts that lead to this thesis.

To all my friends at M.I.T. that made our stay here an enjoyable educational experience in and beyond the classroom; especially to Wendy Weeks and Gaius Nelson.

To Professor Jack Wyman, for his continued interest in my academic experiences.

To Jim and Linda Shirk, for their constant support and encouragement.

To J. Robert Taylor, a special friend that I could always turn to if I needed a word of encouragement.

** This thesis is dedicated to my parents - James and Barbara Gardner **

TABLE OF CONTENTS

Abstract page 2

Acknowledgments page 3

Content page 4

1) **Chapter One - Introduction** page 6
General principles introduced

1.1) Intent page 7
1.2) The Model page 8
1.3) The Tool page 9

2) **Chapter Two - Overview** page 13
Abstraction and synthesis with proposed tool as applied to the design model

2.1) Intent page 13
2.11) Abstraction page 14
2.12) Synthesis page 16

2.2) The Model page 18
2.21) Typology page 18
2.22) Thematic System page 19
2.23) Structure page 20
2.24) Hierarchies page 20
2.25) Classes page 21
2.26) Space page 22

2.3) The Tool page 22

2.4) Summary page 24

3) **Chapter Three - Visual Imagery** page 25
The digital video environment

3.1) Visual Rhetoric page 25.

3.2) Image Access page 26

3.3) Visual Abstraction page 28
3.31) Phase One page 29
3.32) Phase Two page 30
3.33) Video Objects page 33
3.34) Footprints page 34

3.4) Summary page 36

4) Chapter Four - Graphics Editor **page 37**
The Object-Oriented Graphics editor

4.1) Role of the Object-Oriented Graphics Editor page 38

4.2) Concept of an Object-Oriented Graphics Editorpp. page 39

4.3) Behavior in an Object-Oriented Environment page 40

4.4) Creation of Objects page 41

4.5) Hierarchies and Classes page 43

4.6) The Design Model Applied page 46

4.7) Footprints of the Design Process page 46

5) Chapter Five - Synergism **page 48**
The Role of a Central Knowledge Base in compilation

5.1) Windowing Systems page 49

5.2) A Central Knowledge Base page 49

5.3) Objects and Classes in a Knowledge Base page 51
5.31) Assigning behavior and classes to objects stored in the knowledge base page 53
5.32) Exchange of Knowledge page 56
5.33) Control of Classes of Objects page 57

5.4) Footprints in Composition page 57
5.41) Juxtaposition of Footprints in Design page 58
5.42) Footprints as Overlays page 59
5.43) Footprints as Intelligent Adjuncts page 60
5.44) Simulation Using Footprints page 61

5.5) Summary page 62

6) Chapter Six - Design Scenario **page 63**

7) Chapter Seven - Conclusions **page 90**

8) Chapter Eight - Bibliography **page 92**

"Our inspirations assist us when we clear our senses of known solutions and methods. The realization of a yet unthought-of nature and the elements of its form can stimulate an entirely new point of view about everything. Today we talk about technology as though our minds will be surrendered to the machine. Surely the machine is merely a brain which we get, pot luck, from nature. But a mind capable of realization can inspire a new technology and humiliate the current one."

Louis Kahn

INTRODUCTION

Architectural design, like all forms of communication, is based upon a series of signs and symbols. Through this iconography, a design attains its own vocabulary by which to represent its image of reality. This image of reality is very subjective and varies greatly from design to design.

In the design process, knowledge is often represented with diverse media. The designer is able to optimize his description of intended concepts using the unique and distinct character of each medium. Whether the media be drawing, models, or photography, this combination provides the designer with better methods for describing abstract concepts as concrete objects. Each of these media is a tool .

New tools are constantly being introduced. Two such tools are the computer and video. Today, both video and the computer are becoming commonplace components in our society. The integrated capabilities of video, as an investigative

tool, and the computer, as a design tool will be explored through one prototype in this thesis, which assumes each to be an intellectual adjunct in the design process.

1.1 INTENT

In the past, the use of computers in architecture has centered around production. Today, the exploration of the computer as a tool for architects is emphasizing the design process. In the following, we will explore a new form of computational tool, and its potentials for design: the synthesis of computer graphics and digital visual imagery as means for expressing design knowledge.

This thesis describes an additional communication tool for the designer, and develops a prototypical syntactical structure for its vocabulary, using an established model of design. The intent is to explore the advantages and limitations of using the computer and digital visual imagery concurrently.

The new tool will be used to investigate potentials, and strengthen the understanding of abstract notions and concepts in the design process. It would be bold to claim that this tool will help all designers to visualize these concepts. However, it will investigate a new method for both the representation and comprehension of an abstract notion or concept by bridging the gap between the two, exploring a "seamless carpet of knowledge and learning", capable of in-depth investigation, association, and communication of the designer's own personal vision (Lambert 1986).

The proposed design environment includes a graphics editorand digital video images. Both of the forementioned will work in union to form a more comprehensive design tool.

1.2 MODEL

Its is necessary to establish a model of the design process to utilize in a development of this tool. This model will look at design as a systematic approach based on a structured format of rules and constraints. It uses Prof. John Habraken's *Hierachies in Form*. This methodology was chosen for two reasons. First, it is a design methodolgy which lends itself well to computational exploration (Habraken 1983). Near the beginning of the design process the designer develops a set of rules and constraints that become the basis for decision making. Through these rules and constraints a method for analyzing and understanding new and existing environments is developed and a structured format from which to work established. This structured approach aids in the definition of the problems at hand, assisting in the often ambiguous task of "Problem Solving". The concept of designing with constraints was further explored in a PhD Disseratation by Mark Gross titled "Design as Exploration of Constraints" (Gross 1986). Secondly, Prof. Habraken's methodologies reference the single family dwelling, which shall be the building type to explore.

The concept of *Hierarchies in Form* is well developed and documented, especially regarding the single family dwelling. This building type is small enough in scale, and yet complex enough in its potential rules of organization that it will be managable and productive. There is a large body of literature elaborating upon hierarchies and the single family dwelling, yet the concepts and writings are not limited to this typology *. The theory begins at the scale of the urban tissue, and propogates to that of furniture in a room (Habraken 1984), exploring tissue type organization of environmental form.

* (Habraken 1976, 1983, 1984)

1.3 THE TOOL

The architect is a visionary, changing abstract concepts into known, and understood realities. These realities do not exist only in built form: Representations are present in drawings, models, etc. For the designer, an effective communication mode is a necessity. With the help of various media, architects are able to investigate, design and convey their intentions. The paper and pencil are the conventional tools of the architect for representing intentions for built form. If we were to remove this representational media, and not replace it with a comparable substitute, our means for expression could be hampered. We therefore conclude that by limiting the tools of the designer we limit the means for expression of abstract concepts, thus limiting the product. Conversely, additional *meaningful tools* could increase the designers ability to express abstract concepts, improving understanding and thus improve the product.

By introducing an additional tool, we shall add another representational media to aid the designer in the expression of abstract notions. Four primary activities will be available with the introduction of this synergistic tool, which intergrates computer graphics and digital visual imagery into the design process. These activities, visual referencing through a visual database, annotation / sketching, a graphics editor, and compilation of the previous three into a composite image are described below.

Level One - *Visual Database*

The Visual Database allows the designer to view images based upon a category, classification, and/or relationship. Images can range from "Master Specifications", to "window details", to a series of images referencing "churches built between 1860 and 1880, by H.H. Richardson in Boston, Massachusetts". The database is a referencing system for easy access to a

myriad of images, their classifications and relationships.

Level Two - *Annotation / Sketching*

Images are accessed, sorted, and grouped through a classification scheme in the visual database. Annotation and sketching on top of the image can then be used to better illustrate the desired elements of the image, and demonstrate inferences being made about each image. This basic sketch and analysis tool is similar to procedures currently undertaken with a pencil and tracing paper over a chosen drawing or photo. Beyond the basic options for annotation, the designer can also begin to use the computer to help analyze the image, breaking down the various elements into objects. These objects are then categorized by the designer and assigned an established class*, or the designer can create a new class "on the fly". These classes are associated with a hierarchy** and are a relative means for understanding an image, or building.

A class is composed of objects which contain properties of similarity. Each class is assigned a location in the hierarchical framework. For example, the window is in the wall, and the muttons are in the window which is in the wall. Through this example we can begin to understand classes (walls, windows, muttons) and hierarchies (window in wall, mutton in window, etc) with regard to physical location.

Classes are not limited to physical similarities. They can also be grouped by conceptual, or behavior attributes. For example, by manipulating class X, it causes objects A and B to change color from BLUE and GREEN respectively to RED. This physical change in objects A and B occurs due to their binding to class X, which is

of a higher-level in the hierarchy. Therefore, A and B may be categorized in the same class due to their similar behavioral attributes.

With a hierarchy in place, the search process which occurs on a large scale in the visual database can be applied to; each individual image, clusters of objects which may form a class within that image, or a series of images. Therefore, we can query for information within an image just as we queryed for information across multipule images in the visual database. For example, we could query for all the RED objects within an image, and have them displayed in a specified location, implimenting a similar search as that which occurred previously.

Architects do not design line by line, vector by vector. They recognize volumns, sythesizing wholes from parts to give embodiment to their design concepts. In order to adaquately express these concepts, representational tools should conform to these ideals. Establishing a class structure within a hierarchical framework allows the designer to recognize objects, whereby an image becomes a series of objects that describes its inherent vocabulary. This vocabulary, constitutes a method of communication and analysis more directly associated with the process of designing.

Level Three - *Graphics Editor*

The *Annotation / Sketching* level is primarily a subtractive process. The designer is given a complex image, and through analysis elminates the non-essential elements to derive the representation most appropriate for expressing themes and structure within a chosen building type. The *Graphics Editor* acts as an electronic drafting system. The editor is anobject-oriented graphics editor, creating objects

instead of primary elements such as lines. With the editor, the process is one of addition and synthesis. A theme which has been discovered or created, with the help of the typology investigation, is abstracted through transformation. The result in many cases is a more complex object, composed of smaller elements and expressing a desired theme. Class systems can be associated with the objects in a similar fashion as in the Annotation / Sketching level, further elaborating on design as an object oriented process.

In addition to assigning a class to each object, the designer can begin to attribute behavior to objects. This ability to assign behavior introduces objects, and classes of objects, as intellegent adjuncts in the design process, aware of what actions they can and can not perform.

Level Four - *Synergism*

Composition of the aformentioned tools creates a forth level. At this level the designer extracts important elements and phases in the analysis and synthesis process in search of information not attainable by either tool independently. Juxtaposition of these stages against one another is a visual referencing tool, similar to comparing a design of a prairie style house with a house of Frank Lloyd Wright's. With elements in an image capable of retaining knowledge and behavioral characteristics the combination of objects from either tool can lead to more than a visual reference. Composition could suggest the effects that actions taken on an object in one environment have upon a similar object in another: Therefore extending the current use of either digital video or the computer beyond their independent capabilities. An example of this can be found in chapter five.

I have no chair, nor church, nor philosophy
I lead no man to the dinner table or library or exchange.
But each man and each woman of you I lead upon a knoll,
My left hand hooks you around the waist,
My right hand points to landscapes of continents,
and plain public roads.
Not I nor anyone else can travel that road for you,
You must travel it for yourself.

Walt Whitman

CHAPTER TWO

2.1 INTENT

The physical environment encompasses complex configurations of elements. Designers manipulate these physical elements in order to discover the desired configuration best representing their abstract concepts. These abstract concepts are often difficult to represent in concrete physical elements. Each designer visualizes certain things in certain ways, expounding upon various concepts, and experimenting to discover the physical configuration that can best begin to describe the abstract concepts known only to herself. The subject of exploration will be such abstract concepts, concretized to manifest themselves in the built environment, and the methods for expressing and understanding them.

Through the implementation of the tool described in chapter one, the intent of this thesis in three fold in nature: First, discovery of the various levels of abstraction within an image, through image analysis. Understanding the levels of abstraction within an image, or typology, can help the design better understand the potential thematic system which the creator intended. Second, through image synthesis reinterpretation of the levels of abstraction discovered with the analysis process in order to create a thematic system comparable in structure to a certain image, or typology. Third, combination of both image analysis, and image synthesis into composite images, in order to test the validity of image synthesis, and discover a new method of learning for designers.

2.11 ABSTRACTION

Abstraction is the key behind the representation of a non-physical idea, or notion, as a concrete object in the built environment. *Levels of abstraction* refer to the degree of explicit detail which might be present to represent of abstract ideation as a concrete physical element.

The concept of concrete-abstraction has been variously defined in both psychological and philosophical literature. In this instance, concrete refers to the physical reality which exists in the built environment. Abstraction refers to the process of recognizing certain objects or events, while ignoring the irrelevant features (Paivio 1971). In Paivio's discussion of the concept, he describes abstraction in terms of an abstraction ladder. "The 'abstraction ladder' of the general semanticists expresses this kind of definition in terms of levels of abstraction, where higher levels involve increasing

omission of reference to the characteristics of particular objects." (Paivio 1971) Simply stated, abstraction is the process of hiding detail, in order to more easily manage the complexity; the complexity inherent within the natural and built environment.

The general phenomena of abstraction also contains the properties of categories and hierarchy. It refers to categories - not only specific instances - which can be hierarchically organized. Through categories one can build up associations and relations between objects, creating an associative class structure within a given hierarchy. This associative structure spans levels of abstraction, from the highest-level category to the lowest-level category. Images are seen and understood as clusters of elementary objects which contain the properties associated with their specific category. These categories within a hierarchical structure build the framework for abstraction.

Through abstraction, symbolic, or iconic elements can be categorized, allowing for both generalization and detail to be found within an object, or image. Generalization can be described as hiding detail within an image, simplifying the complex, and is one form of abstraction. Within the built environment, objects are often generalized to hide the complexity inherent to them. A house, when viewed from the exterior, appears to be a box with a hip roof. When a designer delves deeper into its contents, another world of complexity is found within the box. In this example, something which might appear to be simple in its composition on the outside is actually quite intricate on the inside.

2.12 SYNTHESIS

Synthesis is the combination of various objects and categories to create a whole and is a step by step procedure, following a series of rules and constraints established by the designer. Abstraction has dealt with generalizing an image, simplifying it into various elements in order to better understand its components and its structure. Through this abstraction, our perception and understanding of the image as a whole can be greatly heightened. Once we understand the objects within an image, and their hierarchical framework, we can create our own instance of that image. If an image is of a specific typology of building, for example a house, we may abstract the components and their relationships, and synthesize our own representation of that typology. Synthesis becomes two-fold in nature; the recognition of various elements, their position in space, and their relationships to other elements, and compilation of them to create a whole. It becomes a process of building from simple elements to more complex objects and relationships. Inherent in the understanding and development of these complex objects, whereby the designer understands the elements and composites them to form the whole, is the creation of a form language and that languages use.

Within synthesis the designer might choose to use the same, or similar, components found in the analysis process. She may use the same structure, grid, or relationships, to build upon what she has already learned. Regardless of the actual number of physical components reused, the analysis of the image through abstraction has unfolded the structure inherent to that image. In the case of the image house, after abstraction the designer should have a clear understanding of the house's elements and their hierarchical framework. By building upon design knowledge attained in this analysis process the designer should be able to recreate a "house" through synthesis. The synthesis of this design knowledge creates an instance of the type house with its own

inherent structure. This discussion of synthesis will be the emphasis of the graphics editor (level three) and composite (level four) respectively.

2.2 THE MODEL

We will view design as a process based upon a structured set of rules and constraints. The issues and terminology described by this model are based on Habraken's Hierarchies in Form. Analysis of a chosen typology uncovers the inherent theme. Through this theme, a designer can recognize a hierarchical framework of objects. These objects are clustered into classes based upon their position in the framework and their behavioral traits.

2.21 TYPOLOGY

Typology, as described by Habraken, is an implicit knowledge or shared image within a culture (Habraken, 1976). For purposes of illustration, we will assume a populist position regarding the framework of this model. In this position, popular concensus guides the designs physical and symbolic qualities. The emphasis is on the use of commonplace elements within a know sociocultural setting, creating a basis from which to develop formal organizing principles. For example, when the word house is stated, it brings to mind images of an implicit typology. Skyscraper, Greek Temple, house; such verbal processes arouse symbolic visual stimuli, bringing to mind a conventional representation of each of these typologies (FIGURE 2.1). These symbolic representations become convention within a culture, with each being described as a typology.

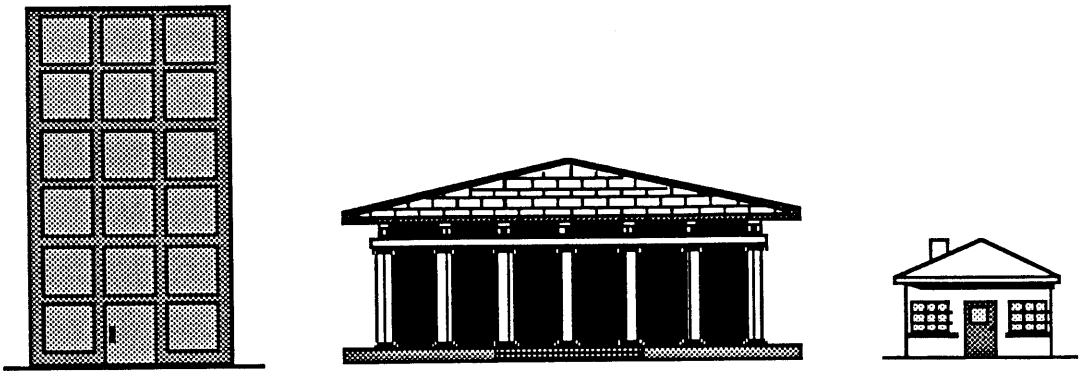


FIGURE 2.1

Throughout this model we shall use the single family house as the typology for exploration. The emphasis on video, described in chapter one, is as an image analysis tool for discovery of themes within the specified typology. It is at this stage that the designer explores an image of a given typology, in search of a set of rules and constraints that became the thematic system for the design. This search occurs by the process of abstraction. Through this search the designer defines the structure of the image.

2.22 THEMATIC SYSTEMS

A theme, or thematic system, is a set of principles upon which to create a vocabulary for design. Within such a theme exists two types of systems; an implicit system and an explicit system. The implicit system is inate to a given typology, and is established through sociocultural forces. The explicit system is the rules and codes that exist within a given typology, brought out into the open. Both of these combined create the theme.

In a thematic system, the designer makes explicit a theme which already existed within a typology. Each system has two parts: the variants and the structure. Variants can be described as the physical elements and objects which make up configurations Structure is the rules and relationships which apply to the variants, or physical elements. Variants are visible, and structure is not. To have a thematic system, both variants and structure must exist. The term thematic within thematic structure describes a set of principles, or vocabulary established with both the variants and the structure.

2.23 STRUCTURE

The basic rules within a typology help create the structure. In the case of the house it is implicit within our cultural image of this typology that it should have a kitchen, bath, and living / sleeping space. We began with the elements or variants within a given typology, later adding organizing principles through abstraction; a class structure and hierarchy. Abstraction, in this sense, may mean the introduction of a field of grids, that help define the structure. The configuration of these rooms, and their relationship to one another will be the subject of exploration within the analysis process. Rules implicit within a given structure are explained through these configurations and relationships.

2.24 HIERARCHIES

Every typology can be described in terms of hierarchies. Variants, or elements, within the typology can be assigned classes, or categories, and ordered in a hierarchically organized meaning system. Individual variants, or clusters of variants, are assigned classes and organized into hierarchical associative structures. This is analogous to the framework of abstraction described in section 2.1.

The theory of Dependency Hierarchies suggests that within the various levels of classes in the hierarchy, the higher-order classes dominate the lower-order classes (Habraken 1984). The classes of higher-order can control the position of lower-order classes. An example of this can be seen when we have walls and furniture. Walls are of a higher-order class than furniture; if the walls are moved, it may create intervention at a lower-order, causing the furniture to move. In this example, the walls are dominant over the furniture, potentially changing the configuration of the furniture.

2.25 CLASSES

The nature of hierarchy is to create the selection of classes, such that classes reside on levels (Habraken 1984). Levels denote the location of a class within the hierarchical framework. The position of classes within the hierarchy suggests dominance and control among classes and levels.

No limitations are imposed upon the number of classes and levels that can exist within a given hierarchy. This aides the generic use of the concept, yet by setting no limits on the number of classes, applications to computational systems are more difficult. One way to describe limitations on the number of classes, or objects, is according to "capacity". Capacity can be described as the ability of a form to allow different variant solutions on lower levels within a given space (Habraken 1982). The larger the capacity, the greater the possible variants within a form, or the space available in that form. In other words, the greater the capacity, the greater the potential for more levels within a hierarchical structure.

2.26 SPACE

A higher level in the hierarchical framework must enclose (geometrically) all lower levels in the classes bound to it. Spaces which are made within existing space can then again be inhabited by new variants and new spaces, limited only by their capacity. It is the property of space that it can contain variants which create other spaces. This expression of space helps strengthen the theory of hierarchies, in which one object is dominated by another, and by yet another.

2.3 THE TOOL

The basic elements of this tool were described in chapter one. The tool is subdivided into four sections; visual database, annotation/sketching, object-oriented graphics editor, and synergism. Its purpose is three fold in nature. First, as a tool for the exploration of a typology, in search of potentials and possible interpretations. At this level the designer is searching for the principles, or theme, behind a given generic architectural form. The understanding of this theme becomes important as a vehicle for learning, aiding in form description through the ability to control complex elements within an architectural idea.

Secondly, the understanding of such themes aids in the process of "design reasoning", whereby a theme must have consistency in order to acquire its own inner logic. Understanding the transformation of simple elements into elaborate organizations of forms and spaces aids in the understanding of this inner logic, and therefore of its application for design. With the image understood, we can now apply functional aspects of design following the chosen theme (Habraken 1982). It is with this application that one utilizes the graphics editor, bringing together all that has been learned into a comprehensive plan.

Thirdly, after analysis has taken place with the visual database, and synthesis has occurred with the graphics editor we explore another direction; synergism. A synergism is defined by Webster's as, "A cooperative action of discrete agencies such that the total effect is greater than the sum of the parts taken separately". In our case, digital video images and computer graphics combined to explore their synergistic qualities. Both media are accepted forms of descriptive media, yet by exploring their potentials united we may make discoveries unattainable with either media separately. The implementation of this occurs using "footprints". Footprints are records of phases in the analysis and synthesis process that are created with the intention of being viewed later. A footprint is more than a simple slide. It retains all the information and knowledge that the original objects possessed. By juxtapositioning the footprints of each tool against the other, at comparable locations in either process, an interesting exchange of knowledge occurs. Comparisons can more easily be seen, and conclusions reached.

The true strength in this concept is that of compilation. Footprints from either, or both tools are composited to discover the implications that actions taken in one environment might have on another. Through composition the designer can recognize meager differences in alternative scenarios, or understand the behavior of a class of objects upon a whole. This melding of information allows objects in either media, video or graphics, to share the information of the other. The implications of this suggest the potential for simulation, which is a viable method for prediction: Simulation techniques can test physical structures based upon their context in a given environment. The elements that compose the physical content are footprints.

2.4 SUMMARY

The model described is closely akin in concept to the theory of abstraction discussed in **2.12** of this chapter. These concepts address hierarchy and classification as elemental design methods. With this method, the designer can divide complex associations into clusters, or classes. By recognizing the associations within a cluster of objects, relationships may be represented at higher levels. The designer can establish a structure for the design based upon rules and constraints.

With a design methodology in place, abstraction and generalization can be seen as a method for understanding the structure of an image. Concreteness is identified as the physical objects that are manifested to express an abstract notion. By understanding these abstract concepts in physical form, more detailed structures can be discovered and implemented. These structures, and the concrete variants that they are composed of, form a descriptive language. In chapters three and four we employ this model, implemented through analysis and synthesis.

"A record, if it is to be useful to science, must be continuously extended, it must be stored, and above all it must be consulted."

Vannevar Bush

CHAPTER THREE

3.1 VISUAL RHETORIC

Vision is inate. Each of us sees the environment in which we live, work, and play, and we are able to report visual experiences. Although the physical configuration may appear different to various individuals, each of us can identify images as being composed of a number of different objects. We know that a tree has a trunk, branches, and leaves, yet recognize that tree as a single entity. This acknowledgment of simple elements in the environment shall be classified as the *primary level of visual understanding.* At this level, recognition of the environment occurs, yet is not fully developed and understood. Visual abstraction is minimal, and awareness of our surroundings is established through context.

Exploration beyond this primary order of visual understanding is undertaken by only a few. To the Visionary, exploration does not stop at the obvious. He delves beyond the surface of visual facts into greater realms of meaning. At this stage visual abstraction begins.

3.2 IMAGE ACCESS

We begin by viewing the obvious, attempting to discover that which is not so obvious. Video is often viewed as an accurate report of reality compressed in time. This media is utilized to directly report and emulate visual details in the environment. With the help of a visual database, images, that may represent a given typology, can be accessed and evaluated. This basic form of evaluation exists at the primary level of visual understanding. Images in the visual database are linked through complex relations. The relational database is the key element and strength.

This tool is a reference guide for accessing images and exploring variations on a typological theme. In the visual database extensive cross-referencing is available through a hierarchical menu structure. Desired images are extracted by the selection of a category, attribute and/or often complex associations. For example, if I was searching for single family detached house how might I go about it? How does anyone recognize an image, or object, as a single family detached house? The simplest way is to see if it contains certain properties. The single family detached house may be described as detached from other buildings **and** not more than three stories high **and** paralleling a street **and** having a yard. **"And"** is the key behind the query process. Each of the physical properties we are attaching to the *house* and searching for is a descriptor; for example three stories high. By querying with more precise descriptors, the extraction process is reduced to only those images that meet the established criteria. This action is similar in concept to a well cataloged slide library. "I want to find houses by Frank Lloyd Wright in Chicago built from 1904 to 1940 that are of brick." "Frank Lloyd Wright" **and** "Chicago" **and** "brick" **and** "built from 1904 to 1940" are all descriptors by which to search.

This referencing and gathering of information utilizes only the basic image access potentials of the visual database. The database is not limited to the information and relations initially programmed. Work by Andrew Bennett addresses the concept of relational thinking as applied to a visual referencing system. It is this work that we will assume to be the bases of image retrieval within the Visual Database.

Bennett's work utilizes a knowledge base system (KES) linked to a videodisk, and / or CD-ROM (Bennett 1987). By attaching a visual database to a knowledge base system, one can understand physical objects contained within a given image and begin to associate intelligence with the search process. This intelligence suggests that attributes or behavioral properties associated with an image, or series of images, can be searched for with the knowledge base. As the query continues, the machine begins to make inferences about relationships through the information it has been receiving. Therefore, by searching for a series of images consistently, and assigning relations to those images, the machine can begin to recognize a pattern, and provide additional information.

Images are retrieved though a series of questions that reduce the query process. As one gets further into the query process, the knowledge base system itself can begin to imply relationships and better aid the user in finding the desired images. A strength of this relational tool is the ability for the designer to create new associations out of new and existing images.

3.3 VISUAL ABSTRACTION

Once a typology is chosen and the desired images are found, the visionary can begin to abstract the structure of an image with image manipulation tools. An instance of the desired image is created and manipulation begins by highlighting the essential elements and features.

In order to further elaborate on the view of abstraction described in chapter 2.1, we will adopt *Donis A. Dondis'* definition of visual abstraction. *"Abstraction, visually, is simplification toward a more intense and distilled meaning ."* (Donis 1973) The removal of extraneous material exposes key elements and features, strengthening the intrinsic meaning of an image. This premise will be accepted for this study and built upon.

Through this abstraction the designer attempts to reach a simplicity whereby she can emphasize the visual elements of primary interest. Patterns in the design become easier to identify, allowing the designer to recognize the intended theme. By dividing the complex image into individual elements one can understand both the complexity and simplicity inherent to the discovered theme.

As an example of this, let us assume that we have a single family detached house typology using the model of hierarchies and classes discussed earlier. The options for simplifying the complex structure of the typology are not unlimited. Yet, no one particular action or sequence of actions can find all the potential variations on a theme. For this reason we shall search for themes in two phases.

3.31 PHASE ONE

The first phase encompasses the whole of the image. At this level analysis does not include extreme simplification. Simplification might destroy the content, thereby not enabling the designer to understand the image as a whole. The method is similar to laying trace paper over a photograph or drawing. Its importance can be understood by analyzing the plan of a small house (FIGURE 3.1), using the concepts of hierarchies and classes described in chapter 2.2. Within the image multiple classes can be identified. In this example we will assign the *site* a class, the *walls* and *columns* another class, and the *furniture* yet another. A class structure represents a configuration of objects and implies a hierarchy within the image of the detached house, or more appropriately, the typology single family detached house. This is further described in the tree structure in FIGURE 3.2. By expressing the diagram as shown we have stated that *walls* and *columns* are dominant over *furniture*, and the class containing *site* is dominant over all classes distinguished. Therefore, any intervention which occurs on the *site* affects all other elements and classes. In this example hierarchy is determined by dominance and control. Movement of objects on a higher level may affect lower levels. The purpose of this analysis is to help the designer understand the structure inherent within classes of objects within a chosen image, or typology.

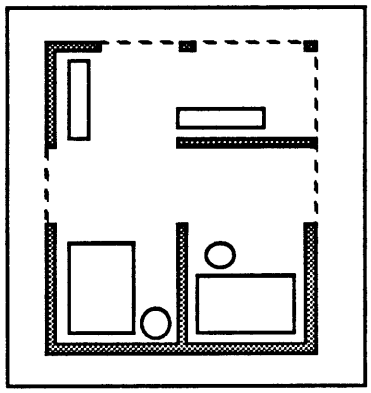


FIGURE 3.1

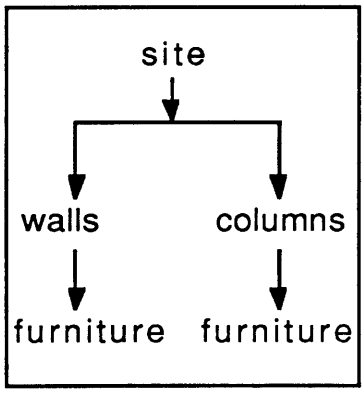


FIGURE 3.2

The hierarchy expressed above is easy to see in the simplistic image given containing a limited class structure. But, one can forsee the difficulties which might arise when applying this holistic approach to a more complex image. In a simple image, the designer can begin to understand the correlation of objects as classes within a hierarchical system, by viewing the totality. The disadvantage to working through only this phase is that the complexity of an image can often distort and obstruct potential themes, limiting the understanding and discovery by the designer. Therefore a second phase is introduced.

3.32 PHASE TWO

The second phase is more complex in nature. At this level we will continue to search for multiple themes through more explicit abstraction. Abstraction breaks down an image into its most essential elements and categories. In this instance, abstraction is used to find detail currently not obvious to the viewer. By the division of the complex image into objects and groups of objects, discovery of multiple themes within various classes of the image can more easily occur.

Habraken expresses this well in the following quote: "Our method of description must allow us to scan the full complexity of the environment by moving down in to ever more detailed specification and up towards even larger abstractions." (Habraken 1984)

As objects are grouped they are also given a class. Each individual class is then extracted from the primary image and an instance of it recreated in another location. At a later time we will composite the instances to again view the whole of the form, with a more comprehensive understanding of the nature of its composition. The intent is to

search for the relations and attributes of objects and classes, through form description. Without breaking down the given image, these relations and attributes might be overlooked.

Once again using the previous image of the house, we subdivide the image into the classes as in phase one. FIGURE 3.32 identifies ClassA, the *site.* FIGURE 3.33 identifies ClassB, containing *walls* and *columns,* and FIGURE 3.34 ClassC of *furniture.* The names given to each class in this example are arbitrary. To distinguish objects further, **types** can be attached to objects within a class. In our definition, type is a simple description of the form. For example, ClassB contains walls and columns which by their class designation signifies their dominance over ClassC of furniture. Yet how do we distinguish walls and columns from each other, and recognize one as controlling the other. We do this by assigning a type to each object in addition to a class. Therefore, walls may be of type *wall,* and columns may be of type *column.*

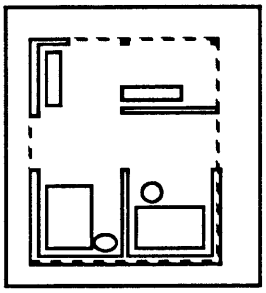


FIGURE 3.31

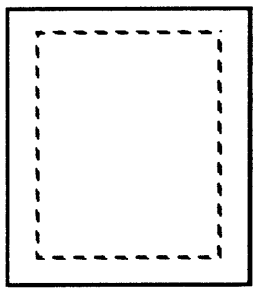


FIGURE 3.32 ClassA

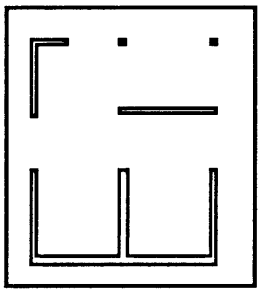


FIGURE 3.33 ClassB

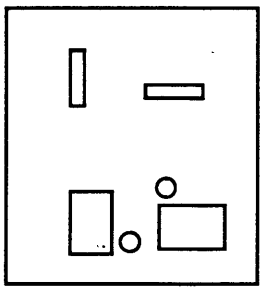


FIGURE 3.34 ClassC

Everything discussed in phase two is very straight forward. By using a simple example containing only three classes, everything could have occurred in the analysis of the image as a whole. To take it further let us now analyze a new instance of a single class of objects.

We will look at ClassB, containing walls and columns. Through analysis we wish to discover the system of organization, or theme, inherent within the class. At this level we could once again simplify the image by subdividing it into classes, repeating the steps previously mentioned. Presently we shall refrain. The image has been adequately simplified.

Using the ability of graphics over video, in an image processor, we begin to explore the organizational system behind ClassB. By superimposing grid-lines on the walls and columns, we explore and analyze zones within the image. Though this analysis we discover a primary grid, which expresses the load-bearing points of the columns and walls, and a secondary grid expressing non-loadbearing elements (FIGURE 3.4). In finding this we would create a new class of objects containing non-loadbearing

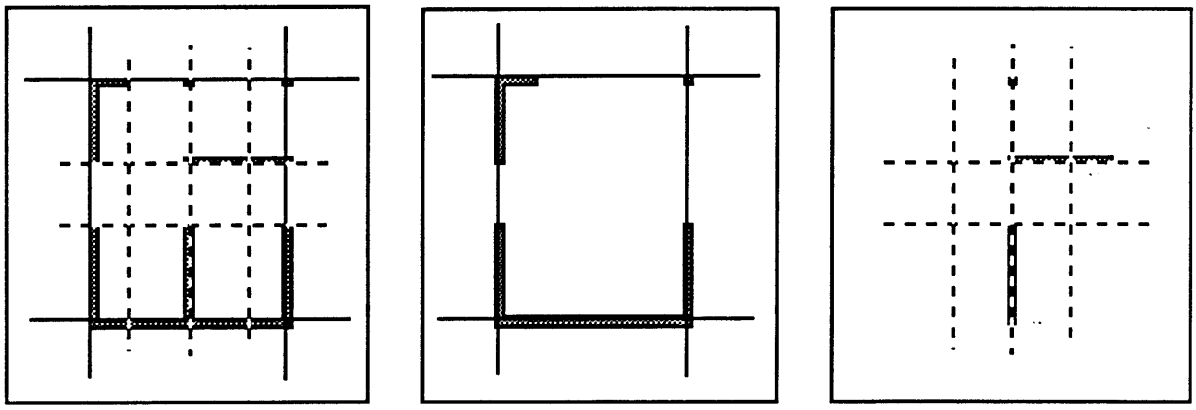


FIGURE 3.4 Primary Grid Secondary Grid

objects and one of load-bearing objects. Note that by descending into a class, we have expressed that class as a configuration of elements, versus the previous phase where elements in a given class did not form coherent configurations: they were considered whole.

3.33 OBJECT DESCRIPTION

The discussion up to now has evolved around objects and their location in a hierarchical system. The framework for classifying objects is built upon Habraken's Hierarchies in Form. Through the analysis process, we disseminate an image to determine its hierarchical framework. When the framework has been established, and classes are assigned, a designer can begin to associate behavior to a class, or objects within a class. "Objects" are entities, or variants, that are define, assigned a class, and can be manipulated. "Behavior" implies that a class of objects can "know" how it will perform under certain conditions.

Classes of objects contain static variables and state variables. Static variables contain information such as an object's size, its color, and its location in the hierarchical framework. State variables imply action, or behavior. For example, the nature of hierarchy suggests that classes of objects of a lower-level can be controlled and manipulated by those of a higher-level. In other words, a lower-level class is "bound" to a higher-level class regarding specific actions. If I have a system of supports in FIGURE 3.33 containing columns and walls, I recognize the supports as being of a higher-level in the hierarchy than the roof. This is recognized because the action of moving the column may cause the roof to collapse. Therefore, the column becomes a state variable, attaching to it the potential behavior that would occur if removed.

By assigning behavior to classes of objects the designer can begin to understand the ramifications that manipulating certain classes would have on other classes. This ability to assign behavioral variables to classes of objects ventures into the realm of object-oriented programming. Object-oriented programming will be further discussed in chapter four.

3.34 FOOTPRINTS

As a designer is exploring potential thematic systems, he may wish to save certain items: classes of objects, graphical annotation over the classes, or both. By creating a footprint, or a file, of the manipulated image, it can be recalled for viewing. A footprint is an instance of chosen objects, or classes of objects, with all the knowledge of the objects retained. Its purpose is storing information that the designer may wish to view, or composite, at a later time. A designer may explore and investigate numerous paths within a given image in search of the thematic system. He can choose a path, follow that path, record footprints of the adventures and potentials, and yet return to the mainstream if the chosen direction did not yield the anticipated introspection. By venturing down a less obvious road, one may find classes, or subclasses of objects categorized that help explain the thematic system. *The footprint acts as a record of important benchmarks on such adventures with their own internal structure based upon the objects from which they were extracted.*

A footprint can be a simple diagram, graphically representing a portion of a thematic system, or can be comprised of complex classes of objects with graphical annotation , emphasizing the structure explicitly. It includes elements a designer feels essential to the understanding and creation of a thematic system, highlighting and recording important objects, and stages in the search and development processes.

The application of a footprint in the digital video environment can be threefold in nature. First, it acts as a recording device of stages along the design process, benchmarking important developments. Second, it creates an easy means for comparison of manipulated images against one another. Juxtaposition of the footprints may aid in understanding the process of transformation, the essential actors, and classes involved. Third, it provides a simple means for testing these transformation, checking consistencies, understanding deviations from the intended path, therefore better understanding the thematic system.

In the process of analysis a designer may discover themes at various levels of the hierarchy of forms. The themes may vary, yet they tie together to form a cohesive whole. By juxtaposition of the primary themes, the composition or the whole becomes easier to visualize. Beyond juxtaposition we consider overlaying footprints over other footprints, creating a sandwich. This action is similar to overlaying layers of trace paper, sketching, reevaluating the product, checking consistency, understanding relationships, testing themes, and exposing the hierarchy of classes of objects. With behavior attached to the various actors, or objects, consequences or certain actions can be more easily understood.

3.4 SUMMARY

Through this analysis, we have magnified a segment of an image in order to bring in more detail and, consequently, greater understanding of the nature of the form. Classes are assigned to objects, and behavior attached to the classes. Intensification of meaning within a discovered theme can exist through injunction and juxtaposition of footprints. Understanding the behavior of classes of objects which are dominated by other classes occurs through the assignment of attributes to the classes. Graphics over video allows the designer to annotate an image, creating a new image, emphasizing discovered themes. Both graphics over video and attachment of behavior to classes of objects, help express contrast and comparison within an image, strengthening understanding of the intended meaning.

The ability to manipulate a digital image, and store the results for later reference, is limited only by the imagination of the user. Viewing and exploring the transformation of simple elements into more complex forms in and of themselves is one potential. Understanding the consequences of actions by assigning attributes to classes is another. Juxtaposition of typologies, injunction, manipulation, distortion, subtraction, addition are all operations which this tool should be capable of undertaking.

Making variations on a theme is the crux of creativity. But it is not some magical, mysterious process that occurs when two indivisible concepts collide; it is a consequence of the divisibility of concepts into already significant subconceptual elements.

Douglas Hofstadter

CHAPTER FOUR

Critical assessment and analysis of existing architypes creates the building block for understanding the intrinsic structure of a typology. Within this structure abstract relationships are translated into physical elements. Conventions which have evolved in architecture over the centuries frame sociocultural expectations, regarding the selection and placement of architectural elements. These expectations reinforce historical examples and the selection and reinterpretation of their elements.

Through analysis, a designer can begin to understand essential elements and the rules of assemblage that constitute a physical representation: individual elements which, when recombined by a the designer, might suggest a different theme, and create a new variation of a conventional typology.

4.1 ROLE OF THE OBJECT-ORIENTED GRAPHICS EDITOR

One often begins the design process through schematic sketches and diagrams. These sketches describe associations between various elements. "Bubble" diagrams, matrices, and/or flow charts are implemented to understand relationships and their complexity. With the help of a graphics editor we begin the development of our own thematic system, building upon knowledge gained in the typological analysis. This thematic system will soon become the bases for decision making and design reasoning.

The chosen model, discussed in chapter two section 2.2, is based on a simple notion of categories, hierarchies and classes. It is not the intent of this graphics editor to restrict design, but, *to open up opportunities for recognizing relationships inherent in complex associations.* With this tool, the process of designing, and the implications of design decisions gain greater visibility This is two-fold in reason. First, assume that the designer has defined the problem and knows her objectives. From this stand point she can critique existing examples, problems and solutions through the analysis process described earlier. Most of this critical information should be gathered prior to entering the graphics editor, as if going to the library and gathering books and xeroxes that represent examples of how this problem, or a similar problem, has been solved in the past. This analysis of examples helps the designer frame the problem, develop objectives, and understand potential opportunities.

Second, we introduce the notion of an object-oriented graphics editor to describe a design method, whereby objects are made aware of their attributes and behavior within the graphics environment. An object can know its size, color, position in the hierarchical structure, and know how to perform certain tasks. For example, if I move a

load-bearing wall in a house I would be notified that it is a structural element and that I can not move it without upsetting the physical structure of the design. The knowledge of such behavior and restrictions allows a system of rules and constraints to be developed and assigned to classes of objects, governing their behavior.

4.2 CONCEPT OF AN OBJECT-ORIENTED GRAPHICS EDITOR

Exactly how a designer moves from subjective concepts to objective attainable requirements (abstract to concrete) in architectural design is somewhat of a mystery. The process of design is often ambiguous: "I like this", "I don't like that", "That is out of proportion", or whatever the aesthetic decision happens to be. Learning, or developing a method of design in terms of such abstract concepts is a difficult and time consuming task. In the following examples we shall adopt Habraken's method of design discussed in chapter two (Habraken 1982). The necessity for using a structured approach will become evident as we proceed.

The *object-oriented,* in "object-oriented graphics editor", refers to designing with structured objects, verses independent elements such as lines and vectors compounded to create objects. This method is closely akin to the way in which a designer transforms abstract concepts into physical realities; a physical reality of elements and volumes. The notion of volumetric design, individual elements composed into a single object, is only one advantage of such an editor, yet it is tantamount to how we conceptualize the built environment.

Typical applications of computerized graphics editors in architectural design do not address structured objects *. Singular entities are drawn such as line, points, vectors.

These entities have little knowledge of themselves, and no knowledge of their behavior. The growing interest in structured objects is leading CADD into an era of the more natural design tool, and interface, by addressing objects as intelligent actors in the design process. Have you ever designed a wall and thought of it as four lines?

4.3 BEHAVIOR IN AN OBJECT-ORIENTED ENVIRONMENT

The structure of an object-oriented system allows the user to assign attributes and characteristics to various objects and classes (COX 1986). The variables and behavior of an object are inextricably bound within it. It becomes an intelligent actor that knows values of chosen variables and knows how to perform certain operations. For example, we may choose to layout walls, whereby some walls are load-bearing and others non-loadbearing. The wall is located at a given x and y coordinate, its color is white and its size is z. Each of these (color, x, y and z) is a static variable attributed to the wall**.

The appearance of these variants is visually the same, but within the load-bearing object is a state variable which indicates that it is structural, and therefore can not be moved without affecting other elements***. If someone attempts to move the wall, a flag could be noted that informs the designer of the implications of his actions. This action attaches behavioral information to objects, and aids in the management of the complexity of a design, by governing the activity that specific classes of objects can and can not perform.

---

* Structured Object refers to an object, or variant, that "knows" its attributes and behavior

** A *static variable* is a variable that is given a value when create but it may be changed: color, size, linetype, etc.

*** *A state variable* is a variable that is attached to an object, or class of objects and contains a behavior attributed to the given objects: movement of this may cause X to occur.

4.4 THE CREATION OF OBJECTS IN THE OBJECT-ORIENTED GRAPHICS EDITOR

Within the object-oriented graphics editor some geometric forms are predefined. The designer will have rectangles, lines, points, etc. to work with. The nature of these objects, and their creation, differs from that of other graphics editors. When a predefined object is chosen, an instance of the predefined object is drawn. An instance refers to a copy of the original object. Attached to this instance is all of the knowledge of the original object, or parent. This propagation of knowledge is referred to as inheritance, and is a distinct characteristic of object-oriented environments. An object inherits the means to perform basic operations such as draw-self, resize-self, moveto, etc **. In addition, each is given its own state. A sample psuedo-code of Graphics Object named "Line" can be seen below:

```
(define Line
    (an-instance-of   Graphics-Object
        (define  :color)
        (define  (draw-me)
                 (move-me  (x  y))
        )
    )
)
```

"Line" is an instance of Graphics-Object. Graphics-Object contains the basic attributes and behavior for all graphics objects: lines, points, rectangles, etc.. A typical Graphics-Object might look something like this:

** The principle design of this editor has been undertaken by James Anderson. From this we will discuss the concept of an object-oriented editor. The graphics editor is currently being amended to include many of the concepts discussed here.

```
(define  Graphics-Object
     (define  x-coordinate  x)
     (define  y-coordinate  y)
     (define  x-size  1)
     (define  y-size  1)
     (define  (resize-me a  b))
     (define  draw-me)
)
```

The Graphics-Object has numerous inate properties. It "knows" how to draw itself, resize itself, its x and y default coordinates, and its size. When we create "Line" we create an instance of Graphics-Object, while attaching other variables and behaviors. Therefore, in addition to "knowing" how to perform all the operations defined in Graphics-Object, "Line" also knows it color and how to move itself. "Color" is considered a variable while "moveme" is considered a behavior.

Both variables and behavior can be inherited from a parent. In the above example, "Graphics-Object" is the parent of "Line", and "Line" inherits some of its characteristics from its parent. In addition to those characteristics inherited, a designer can attach her own variables and behavioral constraints to an object; what operations it can perform, and who it is dependent upon, or is dependent upon it (Cox 1986). A class can be assigned to an object after an instance has been made, or the inherited class changed. One can begin to see the potential that this tool has for managing complexity and change.

The semantics of this system are statically determined. Beyond the predefined, or factory objects*, the user can also create a new object out of existing elements. The designer may take a series of lines and create a polygon. The operation creates a set which contains all the knowledge inherent to each element from which it was built, retaining its knowledge of how to perform certain tasks, and adding to it the inherent

knowledge. In each set, duplicated information can be removed, creating a more cohesive, less cumbersome object. The designer names the object and may then attach other variables to the polygon, such as its class in the hierarchical framework. An instance of the polygon can then be used, passing with it its knowledge of how to do certain things, building up a vocabulary for design. With this ability to create objects, assigning to them names, classes and behaviors, a designer can develop a personalized language of description; *a formal descriptive language with the elements of the vocabulary acting as intelligent adjuncts.*

4.5 HIERARCHIES AND CLASSES IN AN OBJECT-ORIENTED EDITOR

Inheritance as the primary means for specifying the behavior of instances of objects. By understanding inheritance it becomes easy to see how **hierarchy** applies to various classes or objects. As with man, the parent, or parents, supply a child with its identity through inheritance. Properties of the parents are passed to the child, creating the bases for development of that child. When an instance of a graphics object is created, specific characteristics are inate. Because of the nature of propagation, the parent is of a higher-level in a hierarchical structure than the child. Although this is the default structure, a child could be assigned to a class of a different level than that inherited, changing the binding dynamically.

Classes were defined earlier as categories organized in a hierarchical structure. A class can contain multiple objects that contain different variables and behavioral states. This forms a collection of varying elements within any given class. New members can be added to the collection without limit. The size of the class stretches to

*** The term "factory object" is coined from a description of a preprogrammed object by (Cox 1986)

accommodate the newly added objects. Within this collection, or class, procedures can be called to act upon individual objects or upon the entirety. Therefore, the class must know how to perform a set of established operations. It must know how to add and remove elements, and report whether a class contains a member matching the externally supplied criteria. In order for the later option to occur, the operational set within each class must support enumeration*.

The **content** of the class is kept in a data structure, and is considered to be all that is known within a given environment. Content contains all the actions performed on all objects within the environment, including each objects static and state variable. This content is arranged in an ordered list: a linear sequence. As objects are added to a class they are concatenated onto the list of actions that occurred within a given collection. Smaller lists within the overall content describe the information contained in specific objects or classes. For example, I choose to draw a rectangle that is assigned class non-loadbearing. An instance of rectangle is created and assigned *type* 'rect. Within the contents of class non-loadbearing, further subdivision of the list of objects occurs by type. Therefore, although the new instance of rectangle, which represents a wall, is concatenated onto the whole of the content, and concatenated onto the content of the specific class, it is also concatenated into the array of elements of type 'rect, creating lists of lists.

The content acts as a symbolic representation of a class and/or a type; as a file. It is through this symbolic representation that reconstruction of a class, or an object within a class, occurs. This symbolic representation is more commonly referred to as the data

* Enumeration refers to looping over the objects one by one, to perform a certain action .

structure. As you ascend to higher-levels of the hierarchy, the symbolic representation of a class also contains its sub-classes. Through these sub-classes, at any level a designer can understand the dependency of a level upon others because the dependent sub-classes will be included in the **content** of the higher level. The model in chapter two introduced dependencies within the hierarchical framework. The reconstruction of a class with its content also recreates the classes of objects dependent upon that class within the graphics editor. Through this reconstruction, using content, the designer can visualize the affects of intervention that a level, or class, in the hierarchy has upon other classes. (FIGURE 4.1)

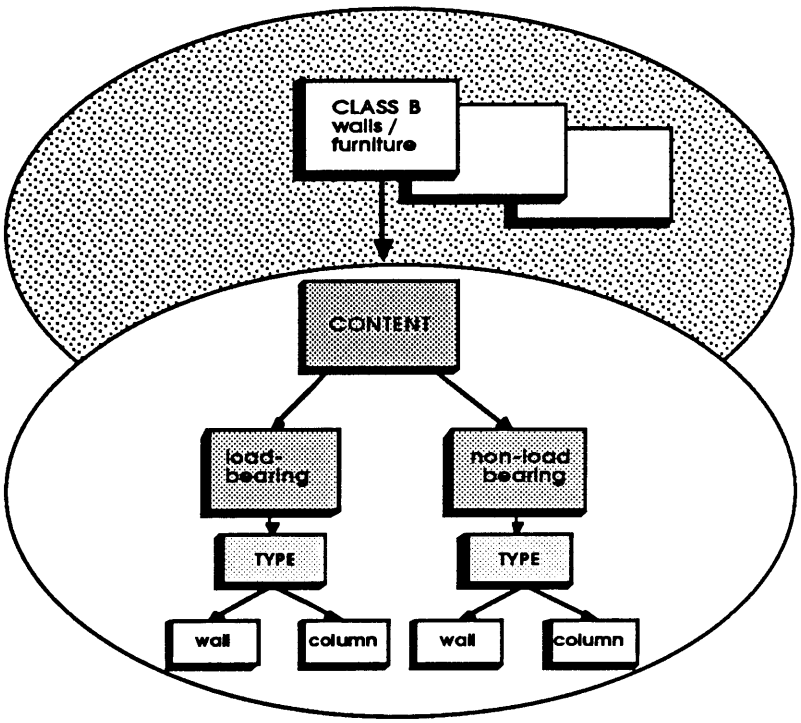


FIGURE 4.1 represents the entities which might exist within a hierarchy, but does not attach or describe the procedures inherent to each object or class.

4.6 THE DESIGN MODEL APPLIED WITH THE OBJECT-ORIENTED GRAPHICS EDITOR

The model of Hierarchies in Form, discussed in chapter two, addresses classes and hierarchies in a conceptual framework comparable, with those in the object-oriented graphics editor. Applications of the model in the graphics editor are concerned with synthesis of the parts into a whole. It is the association of parts to wholes that we examine to discover relationships within composites of objects and classes. The design process can be simplified by breaking down the large problem into smaller problems (abstraction). By viewing design as a synthesis of parts to create a whole, the problem can be viewed as a series of restatements of the objective, through smaller design exercises.

With the help of the object-oriented graphics editor a designer can organize elements, assigning and understanding the behavior of variants, while creating his own vocabulary for discourse. In Chapter Six we view a scenario of how this tool might work: the graphics editor (synthesis) in conjunction with digital video images (analysis).

4.7 FOOTPRINTS OF THE DESIGN PROCESS

In chapter three the concept of footprints was introduced as a tool for recording phases of the exploration process. Its purpose is to compound information for easy retrieval at a later time. Through **content** a designer can maintain a list of all the knowledge of individual objects, or a given class of objects. This knowledge describes all the operations which were performed on the objects, typically in reverse order of their implementation. The ability to recreate the objects at any given time is attained with this data structure, yet it becomes difficult to search for a given point in the process of creation. Therefore, we again introduce footprints as a means for recording these events.

At any chosen time, a designer can create a file, containing the information on specific objects, their states and behaviors. Through this we create a scrapbook of images; objects which have retained their intelligence With this record, we can recreate the process of design, by viewing the chosen images. Beyond this simple record keeping ability, a designer can juxtapose the transformations, further understanding the implications of intervention at different level. Overlaying of the images affords the designer another means for comparison.

We stated earlier that as a footprint is made, it is more than a slide of given objects, it is the recreation of those objects with their attributes and behavior attached. Because of the retention of their knowledge, when compositing images we can see the affects that interventions later in the design process may have had at an earlier stage. For example, if an architect wishes to see the affects that a wall configuration designed earlier and saved as a footprint might have on a later support configuration, he could extract the current configuration and insert the footprint. Since behavior is attached to footprints just as it is attached to other classes of objects, the implications of such an action should be seen.

The potentials of footprints at this level are still ambiguous. Yet the ability to defy time in the design process by viewing the ramifications that certain actions would have if applied earlier in the design process, introduces an interesting use of an anachronism. We shall explore this further in the design scenario in chapter six.

* Variation is achieved by interpretation of the shared image. Change occurs within the constraints of the given system in order to meet the users needs.

Virtually any problem will be easier to solve the more one learns about the context world in which that problem occurs. No matter what one's problem is, provided that it's hard enough, one always gains from learning better ways to learn

Marvin Minsky

CHAPTER FIVE

In chapters three and four we discussed analysis and synthesis with a visual database/image processor and an object-oriented graphics editor respectively. Analysis strengthens the designers understanding of the framework of a given typology, based on images referenced in a visual database. Synthesis takes the form of the implementation of those discovered concepts within an object-oriented graphics editor. Both media strive to reach the same goal: to understand and create a thematic system, by implementing a vocabulary of form description Although they share a vocabulary of form description, each tool operates independent of the other.

Current trends in computational tools are stressing interactivity, where packages are used simultaneously, passing and sharing knowledge. We will explore how digital video and a graphics editor can share knowledge, through the synergism of the two media. The sharing of knowledge will strengthen each tool, and the understanding of the whole, while reducing the amount of repetition present in either media. This investigation will consist of two parts: A visual sharing of knowledge through simultaneous viewing of both tools and their products, and the sharing of attributes and behaviors through a central knowledge base.

5.1 WINDOWING SYSTEMS

simultaneous viewing of elements within either media is possible with a windowing system. In a windowing system a designer can operate an image processor/visual database in established windows, while operating the graphics editor in others. This ability for multi-tasking on the same screen and on the same workstation eliminates the necessity for using multiple terminals to perform various tasks concurrently. Through analysis, objects and classes are discovered and explored. In the graphics editor, those discovered objects, or variations on the objects, are recreated. The simultaneous display allows for visual referencing between the analysis and the synthesis domains. With a windowing system the necessity for multiple computers and screens has been eliminated. This does not address interaction between packages beyond adjacency.

5.2 A CENTRAL KNOWLEDGE BASE

When the designer uses the two tools, a large degree of repetition can occur. Objects discovered in the digital video environment should be able to be used in the graphics environment. Therefore, a method for passing objects, their attributes and behaviors, from one environment to the other is needed. This concept is very simple but the process of creating a data structure that can understand both systems is not trivial. Such a data structure might be constructed from a single knowledge base containing all the objects, classes of objects, and the hierarchical framework from both tools; a **content** composed of all that exists in either environment *. Within the knowledge base is kept all the information that constitutes each object or class of objects, regardless of the environment in which the designer is working at a given time. Residing

* Content was described in chapter four section 4.5

in such a knowledge base would be a filter that extracts the necessary information for working in a specific environment. For example, while working in the graphics editor, a designer may not be able to use the capability for running real time video which is available in the video environment. Therefore, information not applicable to the environment in which the designer is working would be ignored.

As the object is changed in either environment, the original object is changed in the knowledge base. This may pose a problem if the designer wishes to keep the original. In that case the designer has three options: He can create an instance of the object, a child, so the actions taken upon it will not affect the original, or create a footprint of the original to be saved and potentially reused later, or he must have the ability to "undo" his actions in order to return to the state of the original object. FIGURE 5.0

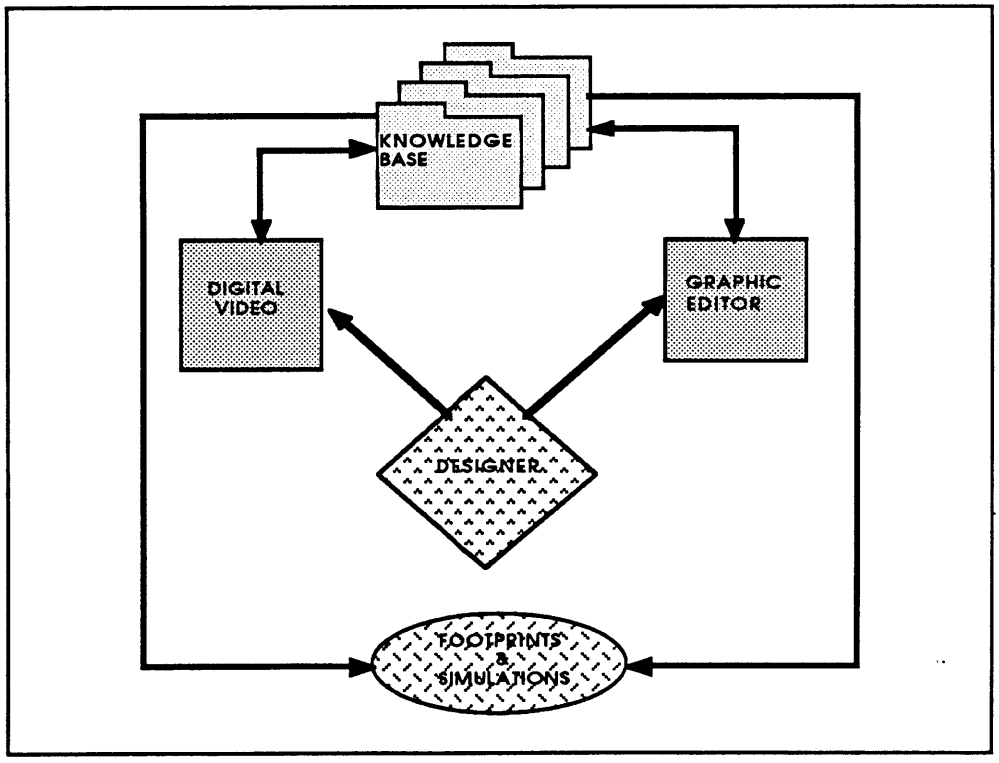


FIGURE 5.0 Diagram of how each tool is connected to the knowledge base.

5.3 OBJECTS AND CLASSES IN A KNOWLEDGE BASE

We currently have two independent environments in which to work; digital video and the graphics editor. Each environment can perform certain operations which the other environment may not be capable of. A designer may begin in the video environment, exploring desired images of typologies, developing a hierarchical framework. Within this framework of classes, dominance can be understood through movement. Manipulation of classes of objects at higher-levels potentially affect lower-levels, causing movement. By binding lower classes in the hierarchy to a higher class, the designer can simulate the behavior of lower classes when intervention occurs at a higher level. When a higher-level class is manipulated, so are the lower-level classes that are bound to it. As this information is discovered, and assignments made, the knowledge is stored in the knowledge base which resides at the top level of both environments.

Objects discovered in the video environment are assigned a class. The objects, and the sets of objects that form a class, are then assigned attributes and behaviors. The subdivision of an established class into small classes forms new classes. Classes of objects can also be grouped by common attributes or behavior forming new classes at higher levels. The knowledge base acts as a warehouse for all of this information. By operating out of one central knowledge base, versus a smaller independent knowledge base for each tool, we have suggested a method by which objects created in one environment can be accessed and understood by another. These interactions can be better understood by viewing FIGURE 5.1.

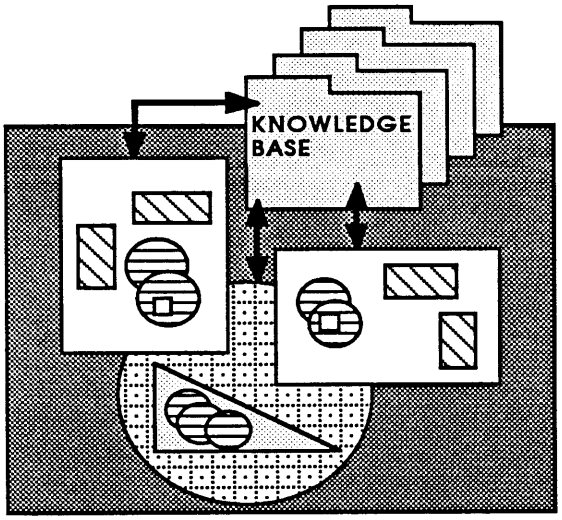


FIGURE 5.1A

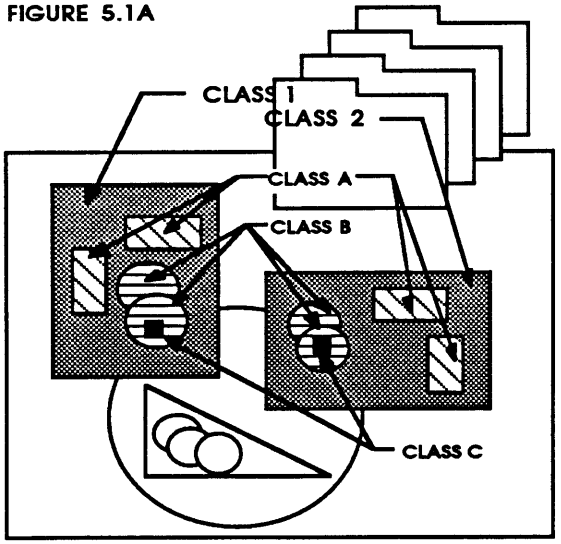


FIGURE 5.1B

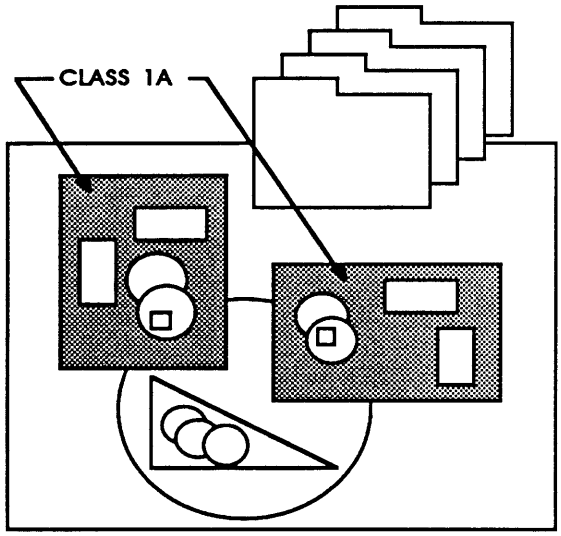


FIGURE 5.1C

FIGURE 5.1 FIGURE 5.1A shows all classes and objects available within the hierarchy. FIGURE 5.1B breaks down the objects by class and type into subclasses. FIGURE 5.1C Groups two classes forming a new class of CLASS 1A.

The environment in which the object-oriented graphics editor resides offers a similar method for handling hierarchies and classes as the video environment. The primary difference is the process by which one arrives at any given hierarchical structure. As stated previously, the designer takes a digital video image and abstracts it into smaller

objects. This can be described as a "top to bottom", or subtractive process. The designer is given the whole and asked to determine the parts of the framework. He is engaged in a "bottom to top", or additive process when he works with a graphics editor. Objects, or variants, which were discovered or created from the analysis process, are reassembled in a new thematic system, to create the whole. During reassembly the designer can access and directly use objects that were found in a video environment, by establishing a central store, or knowledge base, for all the objects in either system, therefore eliminating the necessity to recreate those objects in the graphics editor.

5.31 ASSIGNING BEHAVIOR AND CLASSES TO OBJECTS STORED IN THE KNOWLEDGE BASE

The video environment creates new objects from an existing image, a top to bottom process. When an image is retrieved from the visual database it knows nothing of itself or that which is contained within it. Therefore, the designer must create a structure based on the model of hierarchies in form . Objects are found and assigned the necessary information for description of the chosen typology. Associations are discovered, behavioral traits determined, and a hierarchical class structure built from the objects, and their classes, in order to describe the inherent thematic system. Through this process the designer has developed a language of form description; a vocabulary upon which to build.

The objects manifested from an image become the "super objects" that dictate the language of form description*. These structured objects, and their associated classes, are stored in the knowledge base.

* The concept of the "super object" was described in chapter four

Due to the nature of the central knowledge base, the "super objects" created in the video environment should be able to be used in the graphics environment. In order for this to occur, objects created in either environment must contain some common attributes. In chapter four we described a Graphics-Object which forms the bases for all objects created in that environment. Video objects must retain a similar structure; a structure which has some of the same features that are applied to a Graphics-Object. In FIGURE 5.2 we can see the basic Graphics-Object and a proposed Video-Object juxtaposed. The structure of the two environments, and the operations which they can perform require that the knowledge of an object in a video environment vary from that in a graphics editor, yet the primary structure can be the same.

```
(define Graphics-Object
    (define x-coordinate  x)
    (define y-coordinate  y)
    (define x-size   1)
    (define y-size   1)
    (define (resize-me  a  b))
    (define (draw-me)
)

(define Video-Object
    (define x-coordinate  x)
    (define y-coordinate  y)
    (define (resize-me  a  b))
    (define (reset))
    (define (mask-object))
)
```

FIGURE 5.2 Samples of potential Graphics and Video base objects.

When an object is create, it is created as either a Video-Object, or a Graphics-Object. The basic structure, either Video-Object or Graphics-Object, is determined by the environment in which the designer is working at the time of its creation. This basic

structure does not limits its use to only one environment. Both types of Objects know how to redraw-self, resize-self, etc. In FIGURE 5.2 we can see that these operations are attached to either Object, yet the way that a objects in the video environment redraw-self may vary from the manner in which the objects in the graphics editor performs this task. The knowledge base need not know that there is a difference. Both environments can perform this operation, but the operations differ. Therefore, which operation is performed is dependent upon the environment in which the designer resides. In further elaborating upon this we can assume that there is a hierarchy of operational knowledge attached to the whole of the environment: The knowledge base resides at the top of the hierarchy and is the store for all the information regarding objects, their classes, hierarchies, attributes and behavior. Filters are attached to the knowledge base, which are aware of the operations that can be performed in either environment. At the end of the filter is the specific environment. By choosing to enter either environment, and operate upon specific classes of objects, the objects are passed through a filter, and the information not pertinent to that environment is disregarded. FIGURE 5.3

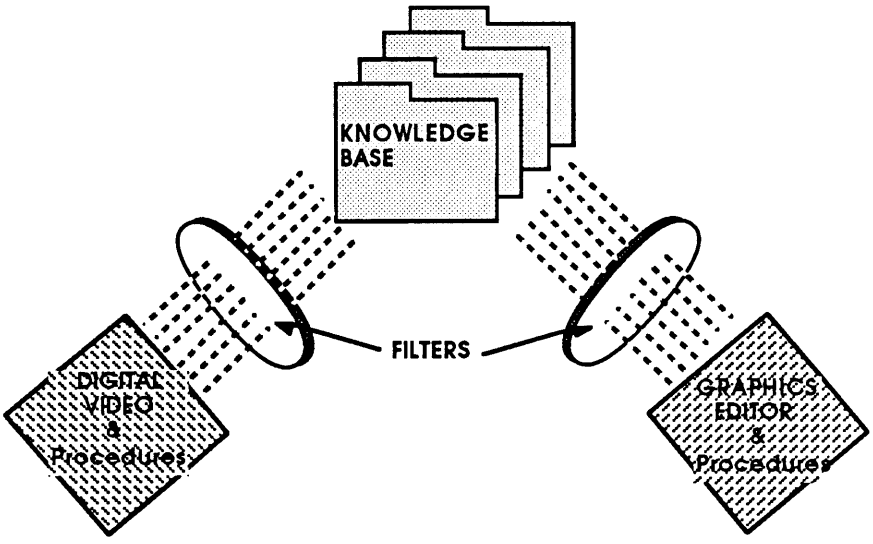


FIGURE 5.3 Diagram for filtering of information from the Knowledge Base into each environment.

5.32 EXCHANGE OF KNOWLEDGE

Super objects can be created in the video environment or the graphics environment, and also used in either. As a super object is created knowledge is attached to it. Additional information can be added to the super object until it is well developed enough as an object to be used to develop the form language. The ability to create super objects allows the designer to develop and build the syntax for the language of form description. It is from these objects that instances are taken. Additional information can be attached to those instances, which build a hierarchical framework. For example, the designer is working in the video environment on the image of a Cape Cod house. In this house is a large fireplace, typical of such a style. The fireplace is extracted as an object in the video environment and attributes and behavior are attached. Such information might be its proportions, and what it is dependent upon. When this object is **first** create, we establish it as a super object, since no other fireplaces exist from which to take an instance.

As we venture into the graphics editor to further explore the typology we may find that the super object of fireplace does not contain all the pertinent information. We know it has a typical location in the Cape Cod house, and therefore attach that information to the super object. We have now created an object that is a formal element in the language of form description used for a Cape Cod house. Each time that a new object *fireplace* is used, and instance is created. This allow the original fireplace to maintain its structure and integrate. Therefore, regardless of the environment in which the super object was conceived, it can be used in either environment.

5.33 CONTROL OF CLASSES OF OBJECTS

Information and knowledge is shared between the two environments through a knowledge base. Changing an object while in either environment effects that object in the knowledge base. This can be advantageous because the designer always has a consistent object regardless of the environment. It can also be quite difficult to recognize that actions taken in one environment are affecting another.

Instances create a copy of a desired object so that the designer will not disturb the original. Footprints also create a copy of an object, or class of objects. Let us now view the advantages and disadvantages of storing the information from both environments in a central knowledge base.

5.4 FOOTPRINTS IN COMPOSITION

In chapters three and four *footprints* were used to compare grouped classes of objects against other groups of objects. These groups constituted a thematic system, or portion thereof. Footprints are diagrams that outline the principle theme, or an important point in the development or exploration of a thematic system.

We view footprints as benchmarks in the design process, similar to saving drawings, photographs, or design and process sketches. Maintaining a record of these benchmarks opens up a new potential for this tool at the levels of digital video, the graphics editor, and their combination: the ability to reevaluate the design process, testing hypothesis, comparing a thematic system discovered in video with that being developed in the graphics editor, and a means to present concepts from their general emphasis at inception to their more detailed final state.

In previous chapters a brief discussion regarding footprints centered around them as a recording, testing and comparison tool. A designer could benchmark different parts of the analysis and synthesis by creating a footprint of the key elements and issues. Later, he could retrieve these footprints, to view transformations made during either process. At this level the tool was primarily a method for recording images and stages in the design and exploration processes.

Just as we assigned variables and behaviors to classes of objects, we should also be able to associate the same with footprints. Objects and classes of objects in a footprint may be hierarchically organized by the designer, added to, or thrown away. During abstraction in the analysis process the designer extracts key elements, while disgarding the unnecessary These levels of abstraction on the "abstraction ladder", can be saved in a file *. Comparison of different levels on the abstraction ladder can later occur, by bringing up the files in various window; viewing them juxtaposed, or overlayed in search of hidden meaning. These files constitute a footprint. We will continue with the concept of footprints and their potentials in the following sections.

5.41 JUXTAPOSITION OF FOOTPRINTS IN DESIGN

Footprints become classes of objects in their own right, with variables and behavior attached. These files are created with the intention of being reviewed at a later time, and act as a medium for sharing information between the two tools; digital video and a graphics editor. With a windowing system, footprints created in the video environment can be juxtaposed against those from the graphics editor.

* The concept of an "abstraction ladder" was described in chapter 2 section 2.11

We have established digital video as an analysis tool and the graphics editor as a synthesis tool. Each tool acts independent of the other. Building from the example given in chapter three, the designer has gone through the analysis process and discovered the objects and classes show in chapter three FIGURE 3.3 and FIGURE 3.4. He understands the thematic system for the structural supports, discovered by overlaying a grid on the load-bearing and non-loadbearing objects in the image, creating a primary and secondary support grid respectively. Moving into the graphics editor, the next step is the creation of his own thematic system. He begins by building up load-bearing and non-loadbearing objects creating the support structure, based on the theme for supports found in FIGURE 3.4 (chapter 3). Continuous layering of classes of objects within the thematic system evolves into the final product. After completion of thematic system in the object-oriented graphics editor the designer may want to check his interpretation with the original architypical image. By creating footprints at similar stages in the analysis and synthesis process a simple juxtaposition of images can occur. This juxtaposition becomes a **visual** reference, comparing the original thematic system with the thematic system developed by the designer.

5.42 FOOTPRINTS AS OVERLAYS

In addition to juxtaposition of footprints, overlaying images in the graphics environment on those in the video environment can potentially lead to undiscovered conclusions. A designer can take a footprint of the grid and support system formulated in the graphics editor, stretch or shrink it, rotate it, and overlay it on the support system discovered in the analysis process. This becomes a check for the implementation of a thematic system, and how transformations made in the graphics editor that did not occur in the original typological example, affect the intent of the original system.

5.43 FOOTPRINTS AS INTELLIGENT ADJUNCTS

Above we juxtaposed and composited footprints from two different media in different "windows" to compare one thematic system against another. The process was not unlike overlaying. For example overlaying a students design of a prairie style house over Frank Lloyd Wright's Robie House, in search of shared properties. This operation is purely visual.

The behavior attached to a class, its knowledge of what actions it can perform, is the same in both media due to the central knowledge base. A footprint consists of more than a visual representation of an image. It retains the attributes, behavioral characteristics, and knowledge of the objects that it was taken from. It is an instance of the desired classes and objects. Many operations that can be performed in either environment, video or graphical, are similar, though not identical. The structure and formulation of footprints allows for discourse between both media. Therefore, we should be able to composite footprints of the two media together, retaining the behavioral properties of each. Classes of objects should recognize other classes, regardless of the original media.

This ability to compile footprints of objects and classes of objects that retain their behavioral characteristics gives us the ability to do simulation. With footprints of both media combined the designer can test ramifications of actions in the graphics environment on those of the digital video.

5.44 SIMULATION USING FOOTPRINTS

A footprint retains the attributes and behavior attached to the objects from which it was taken. Because of this we can simulate behavior of objects, or classes of objects in various environments. For example, objects may be bound physically to another class of objects: for example windows must fit in the cavity of a wall. If the designer creates a window pattern, anda footprint, the footprint containing windows could be compiled with a footprint of walls regardless of the environment in which they were created. A behavioral trait of window states that it must reside within a wall, and move with the wall. Therefore, by moving the wall we should now be able to simulate the movement of the windows. This is a very simple example that begins to illustrate the potentials of simulation, using footprints as the objects of compilation. The key to any simulation, regardless of the complexity, or the environment in which it is operating was well stated by Nicholas Negroponte: "Simulations are no better than their underlying rules, whether the rules are provided by man or machine." (Negroponte 1972)

5.5 SUMMARY

This chapter addresses composition of digital video and the graphics editor into a third tool. The third tool, is viewed as a synergism; "cooperative action of discrete agencies such that the total effect is greater than the sum of the effects taken independently"*. We can understand each tool seperately, and can see examples of their potentials in existing object-oriented graphics editors and image processors. The strength in such tools is not limited to the implementation of each independently. Information and knowledge gained from each can be shared through a central knowledge base which contains all knowledge that exists in either environment. By sharing knowledge we no longer separate analysis and synthesis with such a bold line.

The opportunity for interactively sharing knowledge, recognizing the consequences of actions in one environment on those in another, can create a very powerful design and exploration tool. This form of simulation adds a dynamic potential to the often static process of trying to understand the ramifications of certain actions.

* The definition of SYNERGISM was taken from Webster's New Collegiate Dictionary

No conception or idea could have much use unless it could remain unchanged - and stay in some kind of mental "place" - for long enough for us to find it when we need it. Nor could we ever achieve a goal unless it could persist for long enough. In short, no mind can work without some stable states or memories.

Marvin Minsky

CHAPTER SIX

In the previous chapters we have been discussing a tool for the designer which can be used throughout the entire process of design; from investigation and analysis of existing examples, to design and eventually to presentation. Its unique quality is its proposed method of knowledge representation. An architect addresses complex issues in the process of design: Aesthetic judgment, compliance to existing rules (codes), and functional necessity all govern the final product. In trying to manage this behemoth information, design considerations may get lost, put aside, or bypassed unwillingly. Lines on a sheet of paper represent the concept of the designer, yet retain no knowledge of themselves, or their actions. The projected tool addresses these problems, and proposes a solution: a method whereby the designer can attach knowledge to objects in the design process, forming intelligent actors.

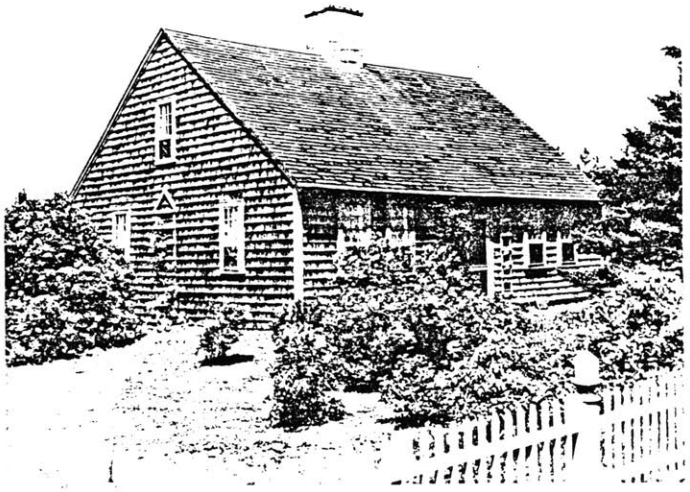
To date it is not fully operational, nevertheless in the following pages we shall step through a scenario of how this tool will work. The chosen typology is the "full Cape Cod house".

6.1 VISUAL DATABASE

The visual database is a referencing system for the retrieval of information. This information is gathered with the help of a knowledge base system and a relational database. Before entering the design process the designer should gather information regarding the chosen topic. We have been asked to design a "full Cape Cod house" and need information on that architectural typology.

With the visual database we query for the chosen topic, requesting information through key words, statements and attributes. We begin by searching for "houses". The query process produces a list of "houses" that currently exist it its library. We continue the search by narrowing the scope. Houses that exist in the Northeast quadrant of the United States, and are no more than two stories tall, and are detached from other dwellings. Through this process we further narrow our options, querying for precise information until we are satisfied with the conclusions, or have found the necessary data.

In our simulation of this level, we are using two sources; visual images of two "typical" full Cape's and a portion of written text that explains some of the history, and vernacular traditions in the design of such a structure. This information can be found below in FIGURE 6.11, FIGURE 6.12 and a segment of text in FIGURE 6.13.



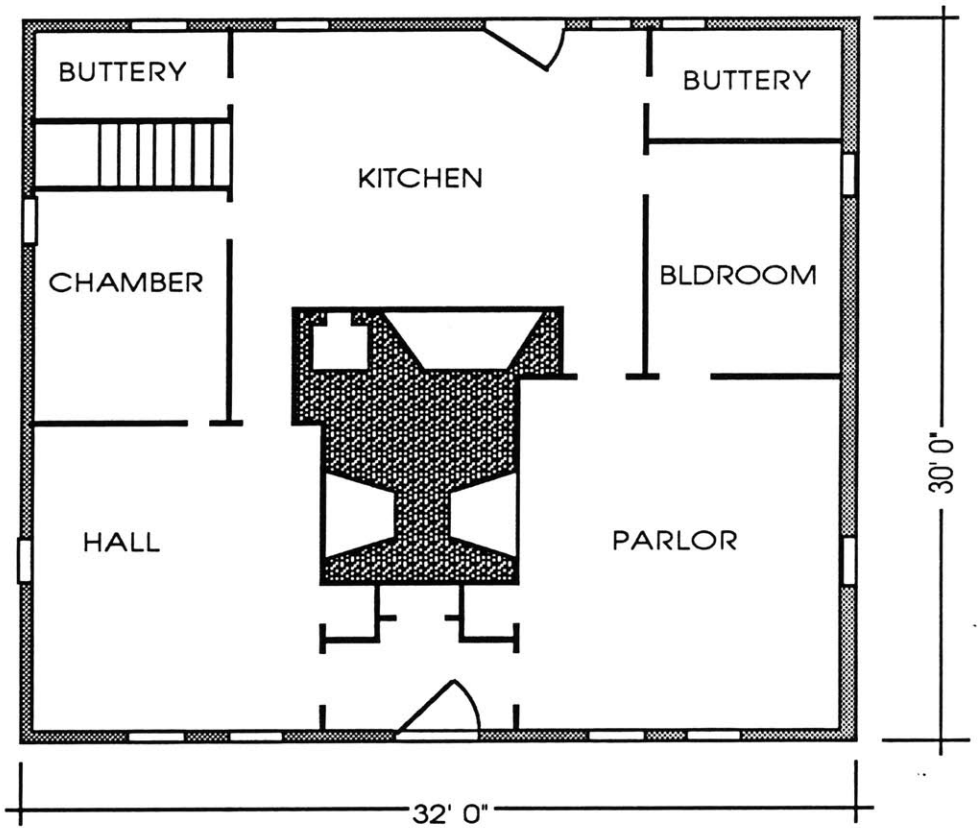
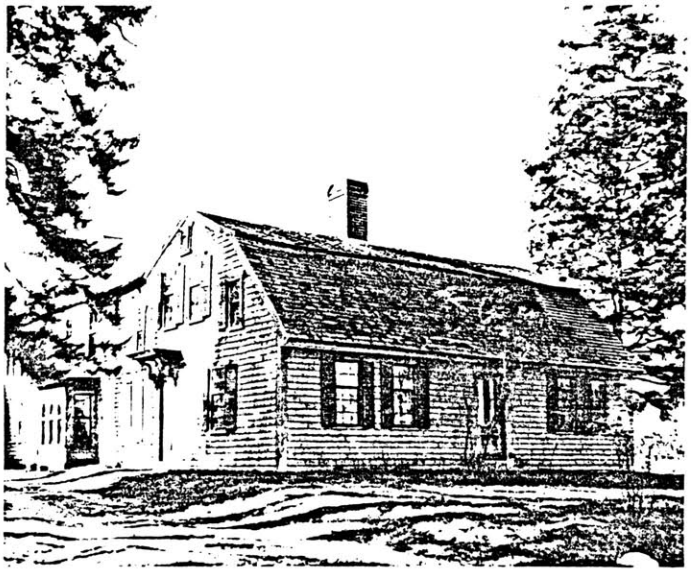


FIGURE 6.11 A full Cape in South Orleans, Mass. built in 1792. Openings were approx. 2 feet wide by 3 feet 10 inches high and are the size normally used in Capes built in early 1700's.



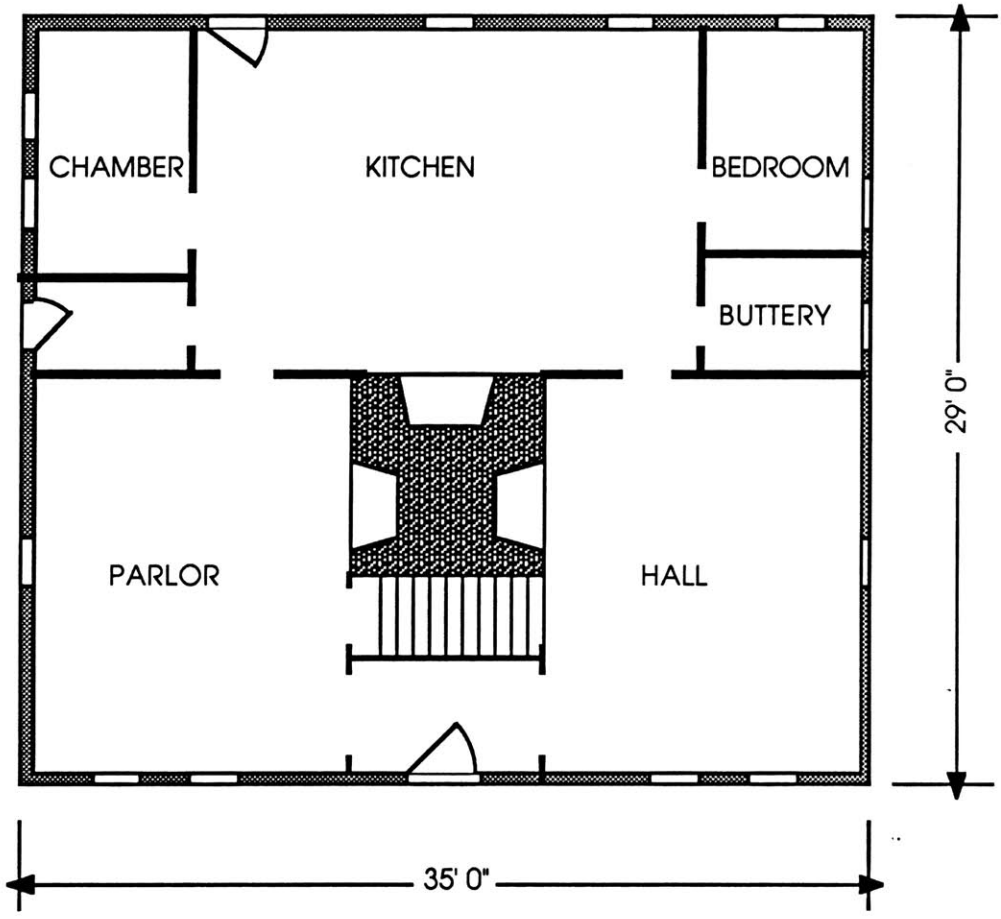


FIGURE 6.12 A full Cape in Lakeville, Mass.

FIGURE 6.13

In addition to images of existing typological examples of the Cape Cod House, we have documentation on the style. The following is a segment of text regarding the typical layout, rules of organization, and general history of the "Cape Cod house". The text is from, "The Cape Cod House: America's Most Popular Home", by Stanley Schuler.

> "Old Cape Cod houses are found everywhere on the Cape but the great majority are concentrated on the Bay side because the settlers were not foolish enough to try coping with the winds, storms and occasional hurricanes that blast the ocean side. As you travel Route 6A from the Cape Cod Canal to Orleans, where the road joins U.S. Route 6, you see Cape Cod houses of every vintage on both sides. And while you are struck by their similarity, you are quickly aware that there are three basic types. According to an early Truro historian, these were designated the 'double house', 'house and a half', and 'house'.
>
> The just plain 'house' is obviously the direct descendant of the early English/Plymouth house. As noted earlier, logic holds that this must have been the first true Cape Cod house; and the fact that it was called simply 'house' supports this belief. Nevertheless, no definite evidence has been uncovered to prove that the three types were not born more or less simultaneously. Down through the years they developed and flourished together.
>
> Modern terminology for the three house types is different. The double house is now called a 'full Cape'; the house and a half is a 'three quarter Cape'; the house is a 'half Cape'. And there are also a quarter cape and a double cape.
>
> The full Cape is to modern Americans everywhere the standard Cape Cod house, although in earliest days it was probably the least prevalent

type because it was the largest. It is roughly 34 to 40 feet long and 28 feet wide. The chimney is precisely centered on the roof above the front door. The door is flanked on both sides by a pair of windows equally spaced. Viewed head on, the picture is one of perfect balance.

The floor plans of the full, three-quarter and half Capes are not so standardized as some people think. True, they have an essential sameness: An experienced burglar would not run into many surprises as he groped his way through a dark house. But no home owner was afraid to deviate from the standard plan if he thought it would produce a dwelling more suited to his family's needs.

In one way, however, the full Capes are almost always alike: When you open the front door, you step into a small entry that is somewhat wider than deep. Ahead of you, though hidden, is a huge fireplace/chimney block, which anchors the house against the gales. To either side of the entry are virtually identical, more or less square rooms, each with its own roughly 3 x 3-foot fireplace, each with two front and one side windows. These were known as the parlor and hall, best room and lodging room, or east and great rooms. The parlor - with the best furniture, including a bed - was used for funerals, weddings, visits by the minister and other import and folk. The hall was a multi-purpose room serving as the master bedroom, a family gathering room and probably a work room.

In the back of the house was the kitchen, or keeping room. It was variously dimensioned, because it was in the part of the house that the owners mainly expressed their own personal preferences; but as a rule it is an oblong room, often quite narrow. Here the family lived, soling up or suffering from the heat from the mammoth fireplace with its beehive oven. Here they cooked, ate, made furniture and worked at everything else under the sun, did whatever close knit families did together. An outside door at the back or side led to the well, privy, other buildings and garden.

At one or both ends of the kitchen were several small rooms. One of these was the buttery used for storage of food, dishes and whatever, for separating milk and churning butter, as also for food preparation. The other was a bedroom. This was the only room in the house called a bedroom; any other rooms used more or less solely for sleeping were chambers. Today this little room has been the borning room, because, besides housing the sick and infirm, it was here that new babies were delivered and nurtured, out of the way but close to their hard-working mothers. It was placed in the kitchen area so it could be kept warm by the ever-burning fire. To the same end, it had only one window; and in a few houses windows were omitted entirely on the theory that light was harmful to infant eyes.

In addition to these two standard room, there were frequently one or two extra small chambers.

Below the buttery, or perhaps the borning room, was a cellar for winter storage of vegetables and summer storage of perishables. Walled with a single tier of special, somewhat wedge-shaped bricks, it was circular to withstand the pressure of the surrounding soil and less than 12 feet across.

The second floor was reached by a stairway that was frequently about as steep as a ship's companionway. This sometimes went up from the kitchen, in which case it was customarily sealed off by a wall and door to prevent heat los s from the kitchen. In other cases, it almost literally climbed up the steeply sloped outside walls of the fireplaces to where they met at the base of the chimney. "

6.2 ANNOTATION AND SKETCHING

Once the necessary information is gathered we begin analysis by abstracting the two plans in FIGURE 6.11 and FIGURE 6.12. We will address only the *plans* in this example for the sake of time and simplicity.

We begin by searching for classes within the image, and their potential dependencies on other classes of objects. In the example from figure 6.11 three classes are easily visible; the structural *support* which includes the exterior walls and large fireplace complex in the center of the house, the *openings,* including doors and windows, and the *walls.* By the extraction of each of these classes, one can more easily understand their structure of composition. FIGUREs 6.21 through 6.24 show this abstraction.

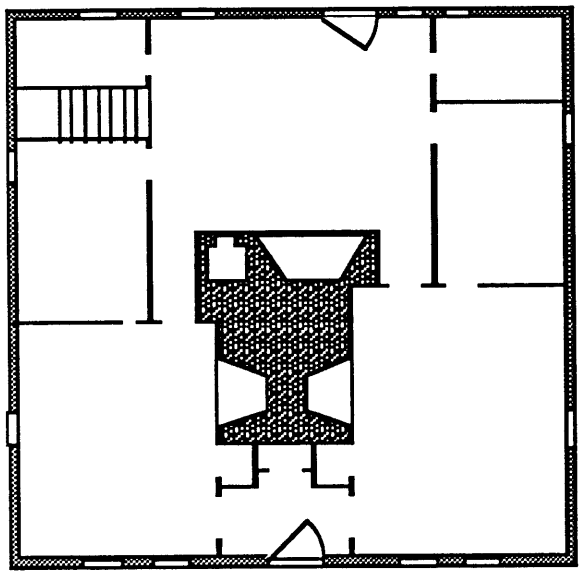


FIGURE 6.21 The whole of the typology.

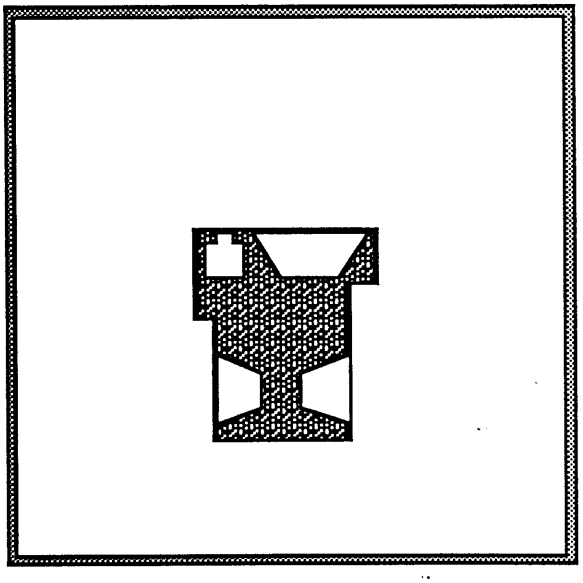


FIGURE 6.22 The structural supports.

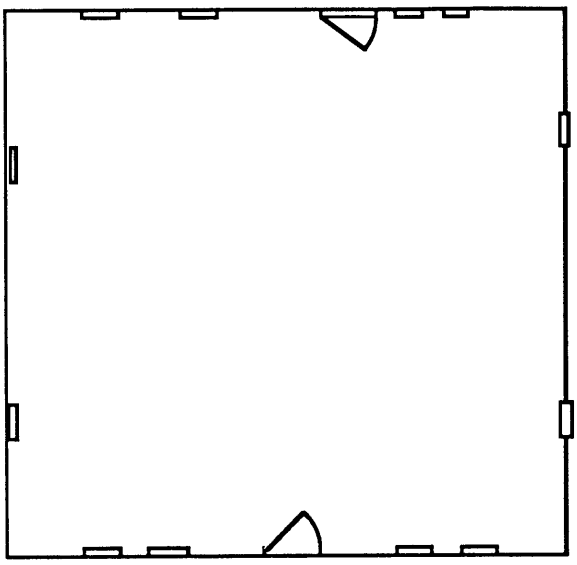


FIGURE 6.23 The Openings of the typology

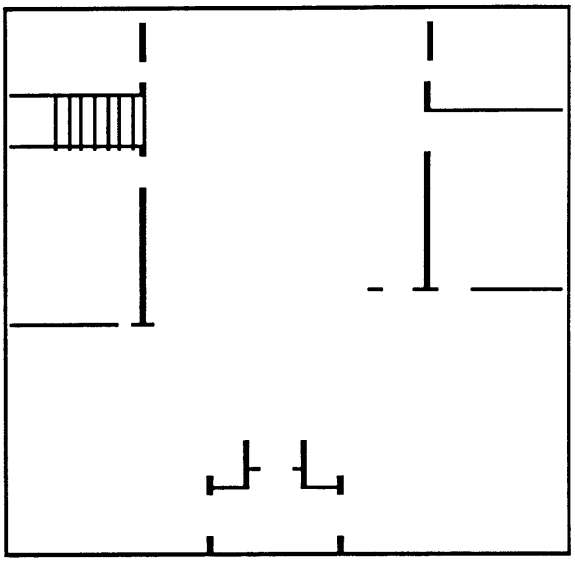


FIGURE 6.24 The walls

After the classes have been determined a hierarchical framework is introduced. This framework suggests dominance of classes of objects over others. FIGURE 6.25 describes the hierarchical structure as it exists at this time.

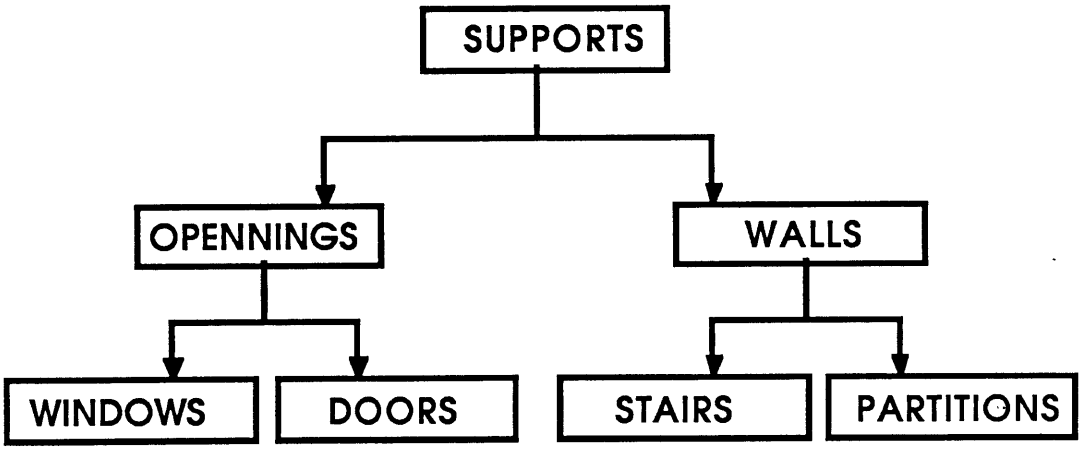


FIGURE 6.25 The hierarchical framework for the house in FIGURE 6.11.

Up to now we have extracted and created objects from an image retrieved from the visual database. Manipulation has been based on extraction and movement. In order to investigate the structure inherent to the "typology" further we implement graphics over the digital video image, and its objects.

The search for a structure to the thematic system begins by overlaying a grid on the openings. In the documentation regarding the "full Cape" house it was stated that two common characteristics are the two windows which flank either side of the front door, and the fireplace chimney that is centered over that door. With the implied grid, the search for the thematic system begins. FIGURE 6.26 demonstrates this. The classes present are the openings and the support structure. At this level the walls are unimportant and therefore were extracted from the image to be manipulated.

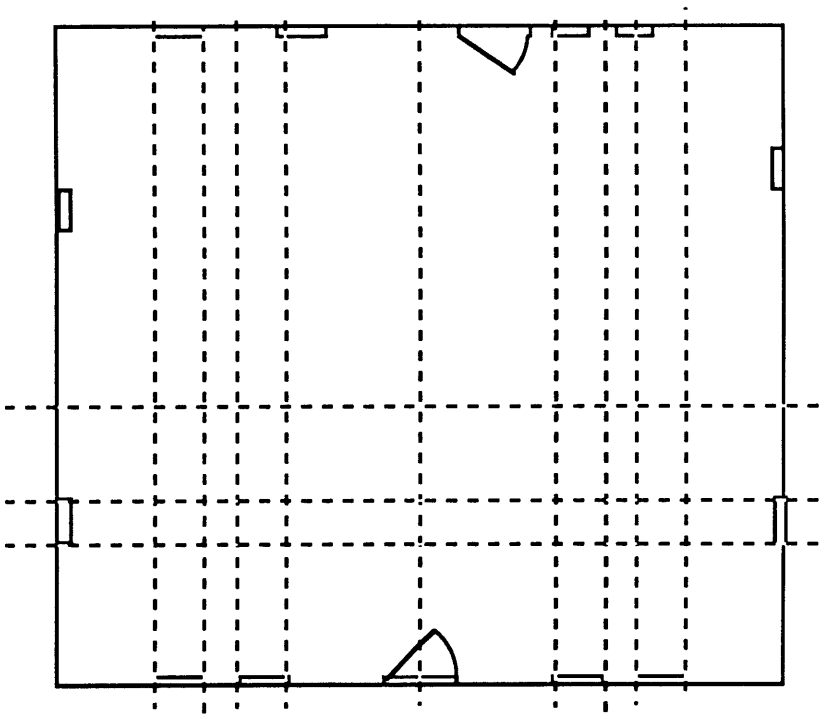


FIGURE 6.26 The grid overlayed on the openings in search of an implicit structure.

After overlaying the grid we can recognize consistencies in the layout of the windows and doors. To describe this we introduce the notion of zones. A zone constitutes the location for particular components of the design based upon specific rules. In FIGURE 6.27 we have applied a grid over the existing image, which we previously assigned a class structure. Once again we address the image as a whole in order to view potential relationships between the applied grid and other variants. Zones help describe the location, and the thematic structure underlying the image of the chosen typology. The examples of the typology are vernacular and were probably not "designed" by an architect. For this reason, variations will occur. The fact that a consistency still exists within the positioning of the zones of the vernacular full Cape further supports the concept of typologies as convention within a given sociocultural group.

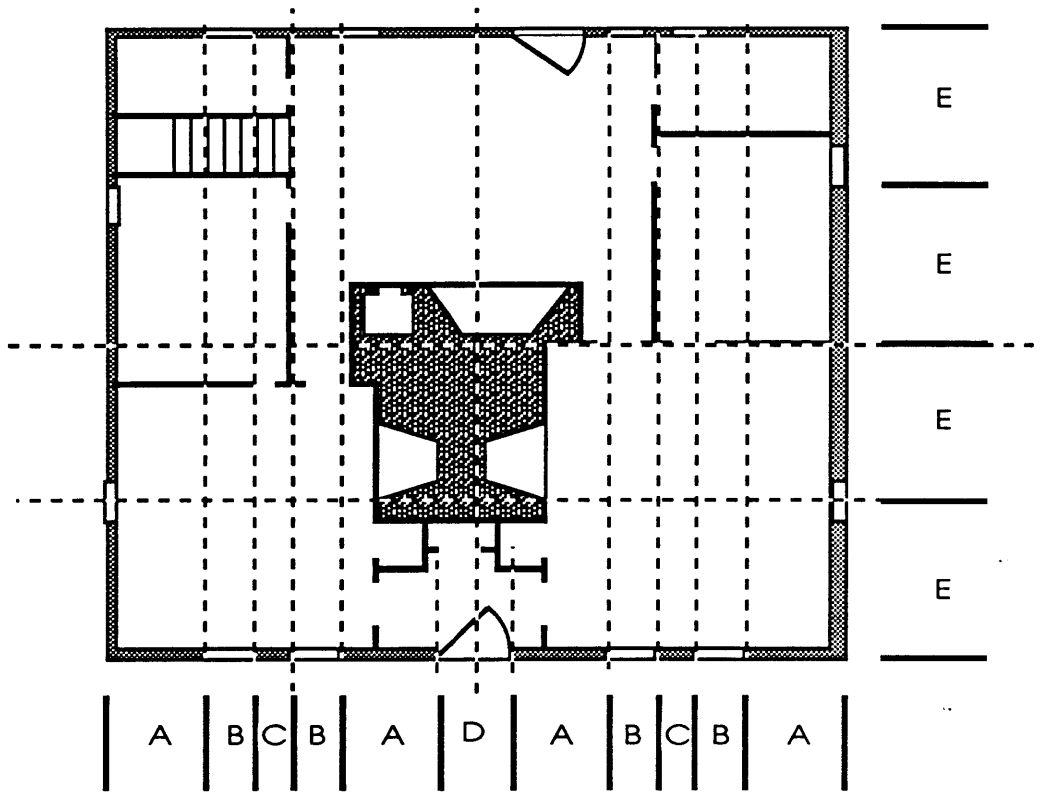


FIGURE 6.27 Describes the zones within the chosen image of the typology. The grid is not ment to represent a scalar system, but a proportional system.

Further investigation regarding the structure of the typology yields other consistencies throughout the chosen examples. FIGURE 6.28 demonstrates this.

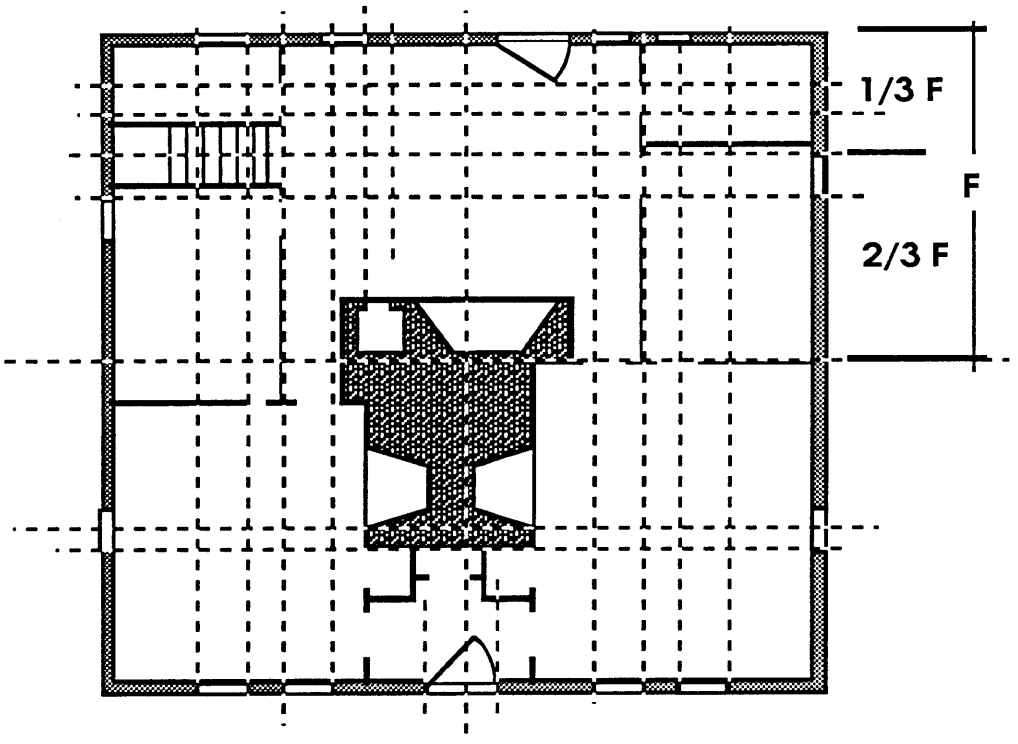


FIGURE 6.27 The proportional system for laying out the back of the full Cape.

We continue this exploration by enacting a similar method or abstraction on the other example of a full Cape house. The following images represent this house at the same stages of investigation through abstraction.

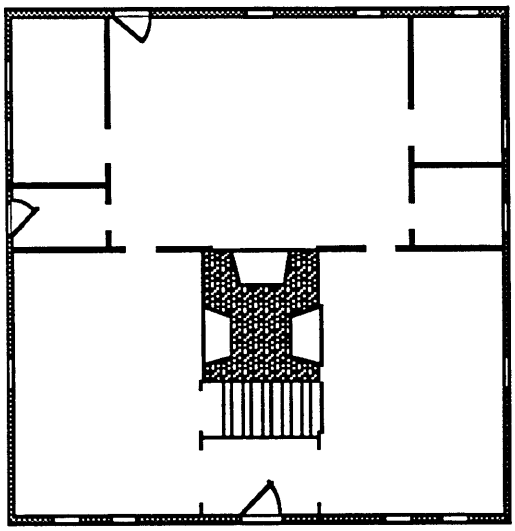


FIGURE 6.28 The full Cape house
See Figure 6.12

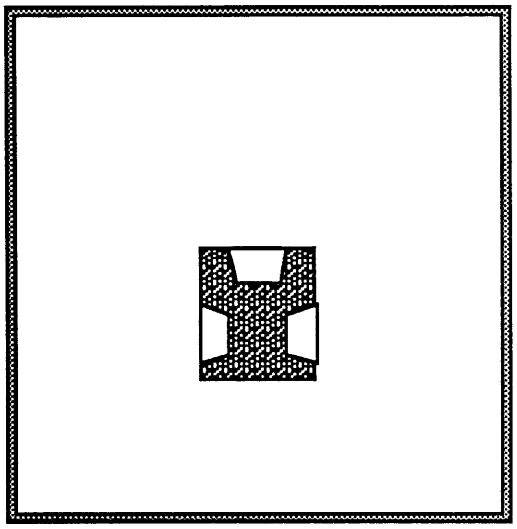


FIGURE 6.29 The support structure

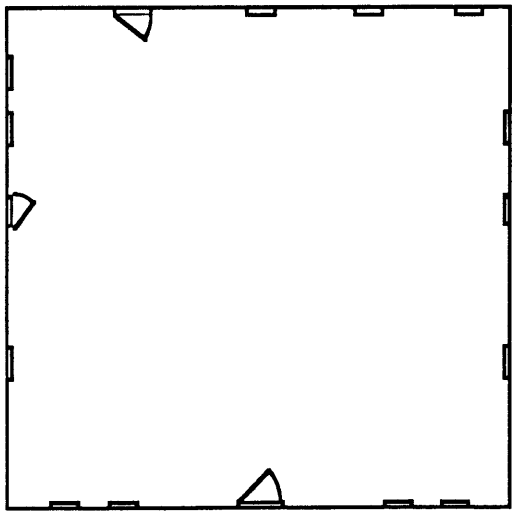


FIGURE 6.210 The Openings

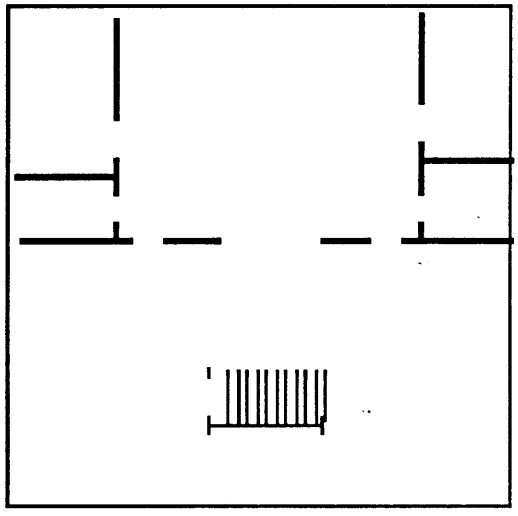


FIGURE 6.211 The walls

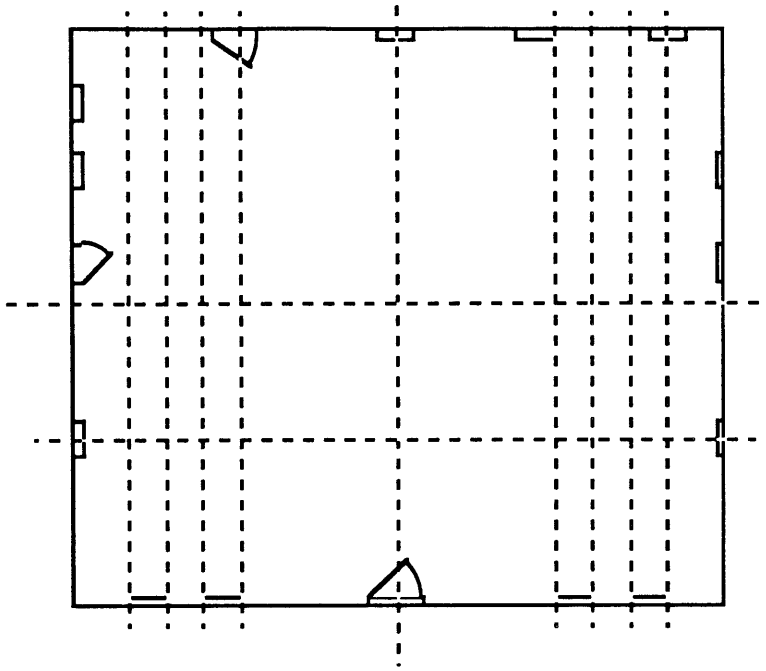


FIGURE 6.212 Overlaying a grid on the Openings helps discover a organizational principle.

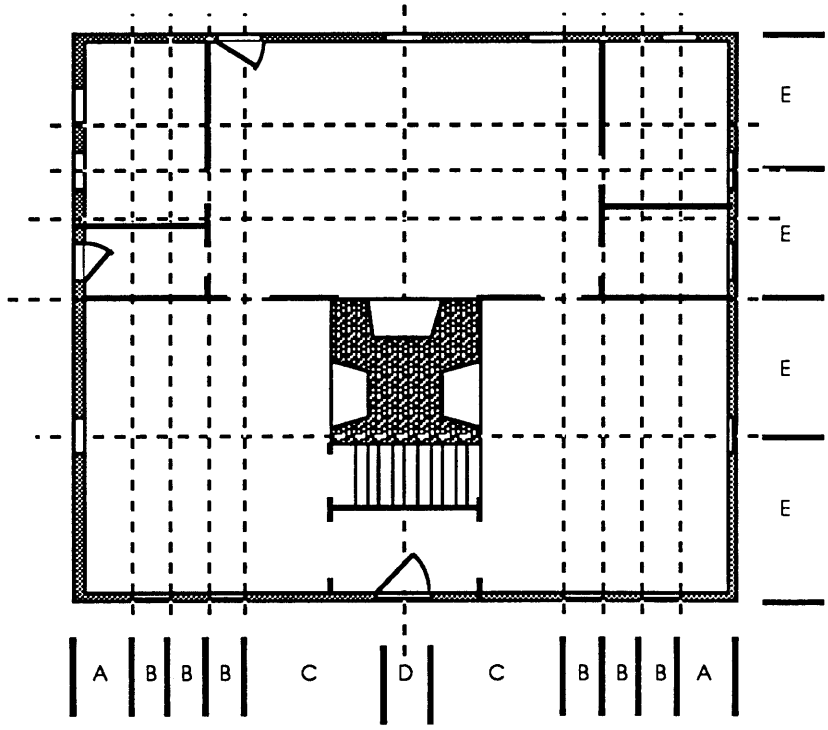


FIGURE 6.213 A proportional grid overlayed on the full Cape. Similar proportional systems can be seen as apply to the previous example in 6.27

6.3 BEHAVIOR OF A TYPOLOGY

The full Cape house is not intended to be pretentious or opulent. Variants composing the thematic system are straight forward and easy to follow. The abstraction of the initial images of the floor plans has been simple. By understanding the different elements within the typology we have begun to understand the implicit theme in the full Cape house. Our exploration has made explicit the implicit structure of the full Cape in order to comprehend the thematic rules of organization and recreate a full Cape house in the traditional style.

Although the overall structure has become clear, variations upon a theme create a richness in a particular typology while maintaining its continuity. These variations can occur only after the designer understands the intrinsic rules and constraints. The geometry overlayed on the previous examples is a tool for understanding the implicit structure of the image. It does not control the environment, but acts as an abstraction of an organization that already exists, or that a designer already has in mind. Variation introduces an important method for deviating from the intended configurations, while operating within the overall umbrella of the thematic system.

In the previous two examples we divided an image into classes of objects within a hierarchical framework (figure 6.25). This abstraction allowed us to understand the components that create the full Cape house and there "typical" methods of composition. The attachment of behavior to an object, or class of objects can reflect the rules and constraints inherent to such a typology. In figures 6.27 and 6.212 we discovered similar proportional systems within both examples that imply the location of openings in a full Cape house. Although we do not want to attach specific dimensions

to these proportional systems, we can extract rules that help determine the location of openings, and apply those rules to any further investigations. By assigning behavior to classes of objects we introduce a method for representing design knowledge.

We know through the examples and text above that a full Cape house "typically" has two windows flanking either side of the front door. We also know that the distance between the two windows is approximately the width of a single window. Because of the location of the class of "openings" in the hierarchical framework we know that they are dependent upon the walls, or that the walls control the location of the openings: Openings must full within the width of the wall. Typically, the designer is responsible for controlling and managing this knowledge, juggling numerous rules and constraints while in the design process. In attaching behavior to these classes of objects we can aid the designer in the management of complex rules, constraints and relationships.

A small example will demonstrate this more clearly. We know that walls control the location of the windows due to their position in the hierarchical framework. If I move the walls the windows must move. I also know that the location of the windows on the front of the house are determined by a simple proportional system. I can attach this information to the classes of objects walls and windows, and watch what might happen.

In the example in figure 6.213 a proportional system is shown. The behavior of the class of walls and the class of windows has been attached to the objects that represent those classes. If I reduce the total width of the house, the proportional system bound to the openings themselves remains consistent due to its behavior. FIGURE 6.31

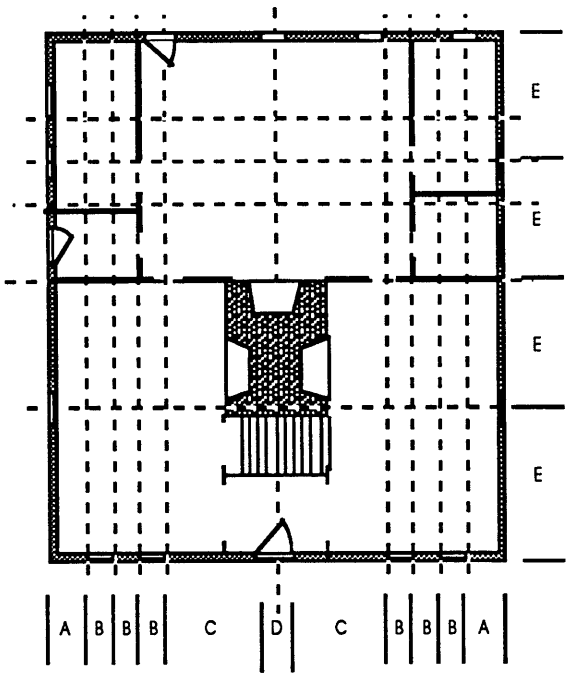


FIGURE 6.31 The full Cape distorted, but maintaining the proportional system established.

We recognize that this example is not truly valid because the constraints upon the typical full Cape state that it is approximately 34 to 40 feet long by 28 feet wide. With this manipulation, we have destroyed the proportional system of the exterior of the building. Therefore, the size of the exterior becomes another constraint , or state, that can be bound to the class of walls, regarding the length of walls running in the x and y direction.

6.4 FOOTPRINTS AS A METHOD OF UNDERSTANDING INTERVENTION

Chapter three introduced the notion of footprints as a means to compare grouped classes of objects against other groups of objects. Each group constitutes a component of a thematic system, and represents a benchmark in the design and exploration process. With footprints we introduce another method to evaluate that which has gone before. When a footprint is created, an instance of an independent object, or class of objects, is created and written to a file. This file represents the image and contains the desired objects. An instance, as described in chapter four, is a copy of an original object that contains all of the knowledge of the original object. Each sample image shown in this chapter might constitute a footprint, except that in the media of paper knowledge can not be attached to an object.

In the following examples we will view footprints as a tool for understanding intervention at various levels in the hierarchy. In a simple case we can juxtapose similar footprints from the two examples to visually compare similarities. FIGURE 6.41 At this level no intelligence is needed on the part of the footprints.

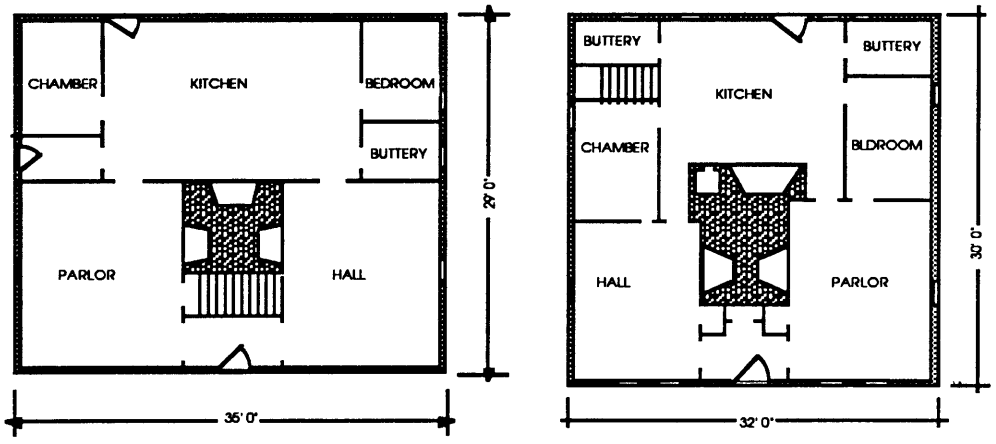


FIGURE 6.41 Juxtapose the two footprints of the sample plans to demonstrate their similarities

Overlaying the chosen footprints is another method for understanding the structure of each. For now we shall leave the study of the affects that intelligence might have on footprints to the final section in this chapter; the section where we compare footprints from the video environment with those from the graphics editor.

6.5 IMPLEMENTATION OF A THEMATIC DESIGN IN THE GRAPHICS EDITOR

In exploring the structure behind the location of the variants, and feeling comfortable with our knowledge, we move into the implementation of our own variation. Through the analysis process we have learned the generic structure of a *full Cape* house. With the graphics editor we shall now implement that which has been learned to create our own variation upon a theme.

We begin using the same hierarchical structure found in the two previous examples. Our dimensions for the exterior are 34 feet by 30 feet; a typical size for a full Cape. The location of the fireplace, and its approximate size are convention so we begin by compositing the exterior and mammoth fireplace. See FIGURE 6.31 The general relationships regarding location of these objects can become part of the objects themselves. Therefore, we know that the fireplace must exist inside the walls of the house and be located approximately in the center. Additional information can be added later.

Next, while referencing the structural layout discovered in the previous examples we create our own variation for the layout of the windows. The proportional system is again typical for this typology. See FIGURE 6.52

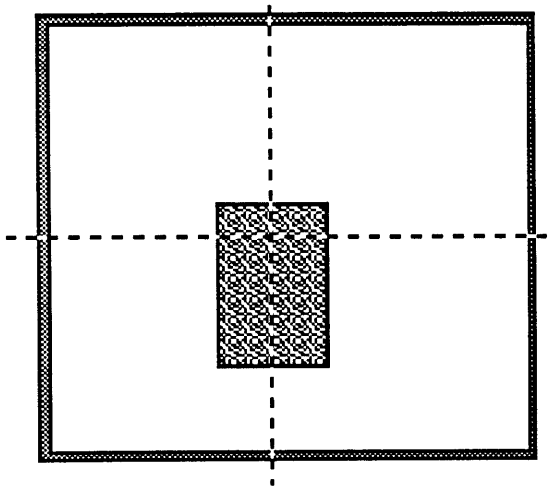


FIGURE 6.51 The typical layout of fireplace for the full Cape.

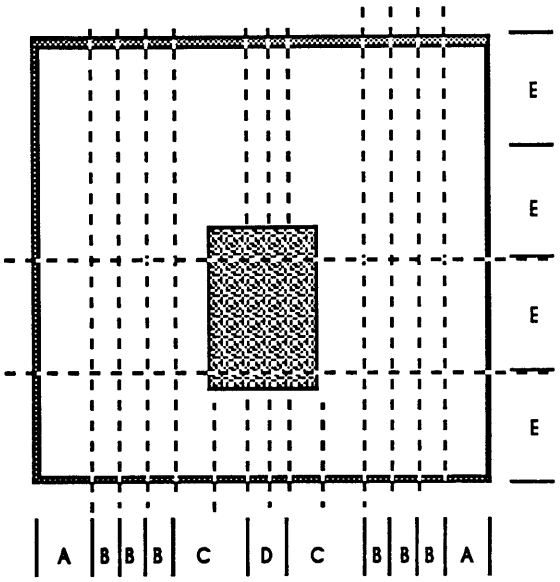


FIGURE 6.52 Overlaying a grid as an organizational system for inserting openings.

We finish the final product with the insertion of walls, again referencing that which is "typical" within the typology of a "full Cape Cod house". See FIGURES 6.53 and 6.54.

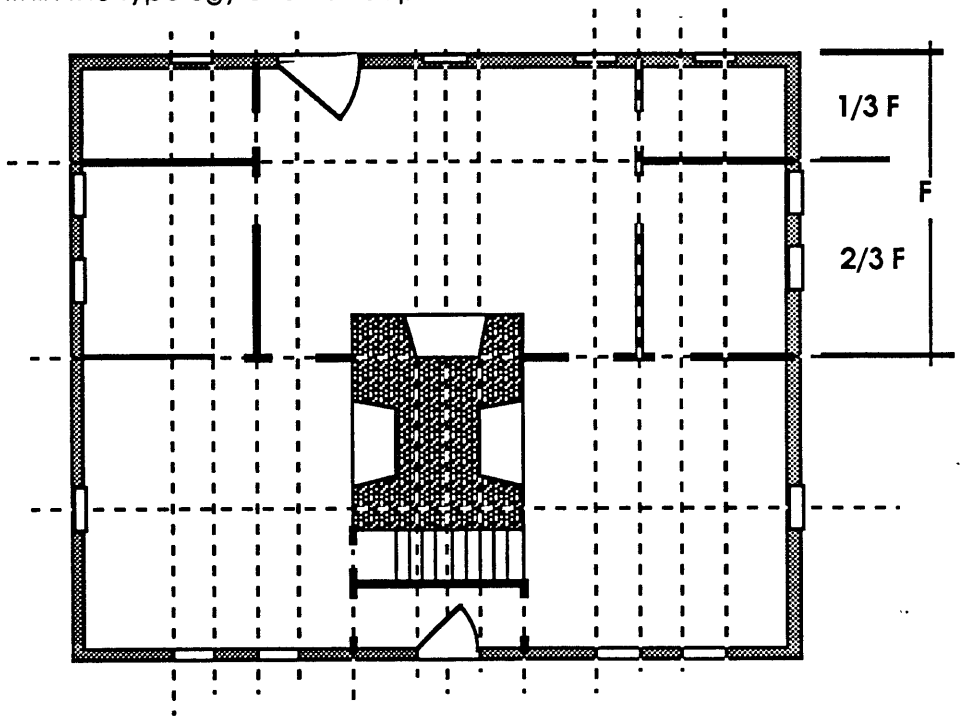


FIGURE 6.53 A proportional system can also be applied to the insertion of walls.

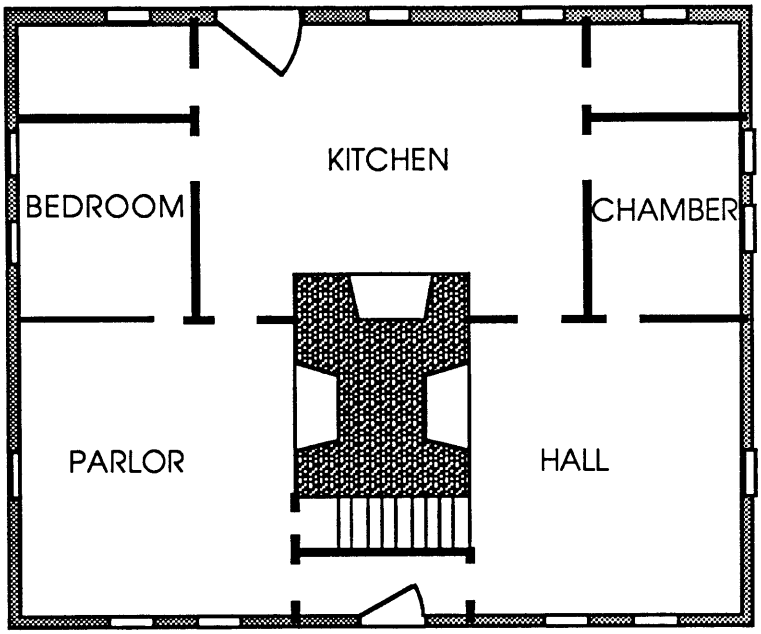


FIGURE 6.54 The final product. A full Cape house in the nature of the inherent vernacular.

6.51 BEHAVIOR OF OBJECTS IN THE MODEL

We have moved quite quickly through the synthesis process with the object-oriented graphics editor. Let us step back a view the process and the potential strength of this tool which may not have been necessary to use is this straight-forward example.

Through the analysis process we discovered a vocabulary of forms common to the *full Cape house*. The primary classes of objects include the support, the walls and the openings. Each of these can again be broken down into subcatagories of fireplace and exterior walls, loadbearing and non-loadbearing, and doors and windows, respectively. Each class of objects has its place in the hierarchical framework.

In analysis we discovered and extracted a vocabulary of form description. We know the approximate size and location of windows, doors, and the fireplace. We also know the general layout of a "typical" full Cape house. The graphics objects which the designer begins with are not necessarily commensurable with what is needed. Therefore, he must create a "super object" from which to make an instance. For example, I know that a window is 2 feet wide by 6 inches deep. Attached to that window is its location in the hierarchical framework. In other words, it must reside in a wall. I might also know its location from the ground, its color, its material, etc. In creating a "super object", or factory object as referred to in chapter four, variables and behavior become inate to that object, and are inherited by each instance thereafter. Additional behavior might suggest that windows that exist in the front of the building exist in pairs, approximately x feet apart. When one window is moved, so too is the other. This behavior does not hold true for all the windows, therefore should be included only in the windows that it does affect. By adding this behavior to an instance of an object we have customized specific instances depending upon their location. Sample psuedo-codes of the window, and the special case window are shown below. The base Graphics - Objects is the same one referenced in section 4.4 of chapter four.

```
(define Window
    (instance-of  Graphics-Object
        (define :material wood)
        (define (resize-me 2'0" 6"))
        (define (move-to   x  y  (but only within the wall))
    )
)
```

In this sample, Window knows that it is wood, its size is 2feet wide by 6inches deep, and that if it moves it must move within the wall. Wood, and 2feet wide by 6inches deep are static variables. "Must move within the wall" is a state variable, expressing the consequences of the specific action .

In addition to the information that exists for the object Window, addition information is added to the windows at the front of the house.

```
(define Front-Window
    (instance-of  Window
        (define  (pair-of-Window  (x-coord  y-coord  direction)))
    )
)
```

A front window has inherited all the knowledge of its parent, Window, who has inherited its knowledge from Graphics-Object. In addition, Front-Window knows that it has two windows which are bound together proportionally. This information was learned while in the analysis process and applied to the class of objects containing Front-Window for synthesis.

By understanding the thematic system of the chosen typology, the same process of developing objects could happen for each class, until the necessary vocabulary exists. This vocabulary creates a library of objects and classes that know their general syntax, and can thereby act intelligently. The implications of intelligent objects suggests that the machine would be capable of managing complexity; but its actions are a direct response of user input.

6.52 FOOTPRINTS IN THE IMPLEMENTATION OF THE MODEL

Throughout this example the transitions from each stage in the exploration and design process have been very logical and obvious. Adding behavioral traits to objects and classes of objects introduces a method for controlling complexity while operating within the rules and constraints of a typology. In section 6.3 footprints were applied to the analysis process as a means for testing conclusions against other examples, while recording benchmarks in the analysis process. Footprints can also be use in the graphics editor.

We assume that the applications discussed in section 6.3 of analysis also hold true for synthesis. Therefore, we shall venture further into their potentials. Behavior attached to a graphics object is recorded in a footprint. A footprint may comprise objects from a single class, or from multiple classes. As we proceed through the synthesis we venture down paths in search of solutions at various levels of the design. A design exercise can be described as a series of smaller exercises compiled together to form the whole. Footprints are created by the designer at different points within the process, recording the stages that evolved.

In the design process, variations of the smaller design exercises occur in search of the best solution. By recording these variations we can compare one solution against another, testing the various alternatives in relation to the whole. Footprints are the method for recording this information. Each footprint contains the behavior of the objects that it was taken from. By compositing alternative footprints within a given structure, the ramifications of their composition should become obvious.

For example, we assume the designer has two options for the layout of walls at the back section of the full Cape house. He has chosen one, but would like to be able to evaluate the effects of the other. Footprints were made of both alternatives of the class of objects containing "walls". By extracting the existing class of walls from that area of the building, we can insert the other option and view the effects it might have in an existing context.

We assume that in creating the "super objects" for windows and walls that a behavior was attached claiming one could not intersect the other. Yet, as we insert the alternative for walls we find that several of the walls intersect the class of objects of "windows". At this point, the implications of this action would be reported back to the designer, notifying them that they have deviated from the preestablished structure. This compilation of alternative parts of the design solution can help the designer visualize options and their ramifications clearly. What would happen if we could apply this concept to both media simultaneously?

6.6 FOOTPRINTS AS A SYNERGISM

In creating a synergism we wish to share knowledge between the two independent media; digital video and the object-oriented graphics editor. We suggested in chapter five a general data structure which might support this concept and encompasses both media.

We demonstrated in the design exercise that information learned within the analysis process is often applied to the synthesis process. The knowledge base eliminate the need for repetition of objects by allowing for the sharing of information. Footprints allow a designer to record important phases in the design process. With footprints, the design can simulate the behavior of a given design with alternative solutions. The language between the objects in the digital video environment and those in the graphics environment are compatible, and discourse between the two media is feasible.

Figures 6.11 and 6.12 show examples of the "full Cape house". Through abstraction we have divided the images into objects and assigned the objects classes. The hierarchical framework, and classes of objects discovered in analysis are similar to those implemented in synthesis with the graphics editor. The central knowledge base keeps track of actions that occur in either environment, thereby allowing the designer to view the implications that actions in one environment have upon those in the other. By compiling footprints, a designer can actually create a new composition out of classes of objects from either environment. He may wish to test his layout of windows by applying it to a "typical" example of the "full Cape", and better understand the implications of his actions. The behavior of each class of objects is retained within a

footprint. Therefore, compositing an image representing objects, or classes of objects, from one environment into another may yield unexpected behavior, assuming that the hierarchy in one environment is the similar to that of the other. The potentials for simulation ineither or both environments introduce a new dimension into design, whereby architects can simulate the affects incurred by specific actions.

In addition to passing behavior through footprints, we can also pass images that contain no behavioral traits. For example, we can combine an image of a site plan with the actual plan of the house that would occupy that site, and manipulate either until the composition is correct. This is purely visual, but purely visual is still an important factor in design!

As soon as the designer furnishes the machine with instructions for finding a method of solution, the authorship of the results becomes ambiguous.

Nicholas Negroponte

CHAPTER SEVEN - CONCLUSIONS

An architect combines various media during the design process in order to more clearly express her concepts in the desired configurations. Through these media, abstract concepts and notions are made concrete by evoking a language of form description. With the tool described in this thesis a designer is able to more easily discover common forms, and their syntax. These forms are found through abstraction and constitute the objects that create the whole.

The ability to attach a structure to an object containing information about itself and its location in the hierarchy, including behavior, introduces objects as intelligent adjuncts, not simply lines on a piece of paper that hold no meaning to anyone other than their creator. These objects are structured objects. Structured objects, and their location in a hierarchical framework, aid the designer in the management of complexity in the design process by helping to manage themselves. Objects can be aware of the environment around them, and the activity occurring in that environment. By attaching such knowledge to objects, the designer can view the implications of her actions dynamically.

The bold line which typically exists between these two independent environments, video and graphics, has been diminished, and interactivity occurs in the computational system with either tool as it does in the designer's creative conscience. The two tools which typically operate separately and independent from one another can now act as a team, with their goal to aid the designer during his quest to translate abstract concepts in physical realities.

In addition, because of this tools interactive nature, simulation can occur, introducing an interactive, instantaneous process for discovering implications that various design alternatives might have in a given setting.

The proposed tool does not claim to replace any existing tools or media. But its introduction can alleviate some of the more time consuming, laborious tasks of the organization of design knowledge, while exposing unseen configurations. Its ability manage complexity frees the architect to "create" versus spending most of his time managing.

BIBLIOGRAPHY

(Anderson 1964) Anderson, Alan Ross
Minds and Machines
Prentic Hall, 1964

(Barthes 1977) Barthes, Roland
Image, Music, Text
Hill and Wang, New York 1977

(Bennett 1987) The Structure of Conservation:
Experiments in Representing knowledge for
Arid Lands Design
Masters of Architecture Thesis
M.I.T. 1987

(Dawson 1987) Benjamin M. Dawson
"Introduction to Image Processing Algorithems"
BYTE, March, 1987

(Cox 1986) Cox, Brad J.
Object Oriented Progremming: An Evolutionary Approach
Addison-Welsely 1986

(Dondis 1973) Dondis, Donis A.
A Primar for Visual Literacy
M.I.T. Press, 1973

(Van Dam 1984) Foley, J.D. Van Dam
Fundamentals of Interactive Computer Graphics
Addison-Wesley 1984

(Foster 1985) Foster, Tom
The Information Technology Revolution
M.I.T. Press 1985

(Gross 1986) Gross, Mark
Design as Elploration of Constraints
PhD Thesis, M.I.T. Dept. of Architecture
Febuary, 1986

(Habraken 1983) Habraken, N. John
Writing Form
M.I.T. Press 1983

(Habraken 1984) Habraken, N. John
Notes on Hierarchies in Form
Working Paper, 1984

(Habraken 1976) Habraken, N. John
Variations
M.I.T. Press, 1976

(Hofstadter 1979) Hofstadter, Douglas R.
Godel, Escher, Bach: An Eternal Golden Braid
Basic Books Inc, New York 1979

(Jurgensen 1986) Jurgensen, Peter
The Conceptual Design for Architectural Form:
A Performance Spec for a Computer System
SMArch Thesis, M.I.T. Dept of Architecture
May, 1986

(Koberg 1973) Koberg, Don & Jim Bagnall
The Universal Traveler
William Kaufmann, Inc 1973

(Lambert 1986) Lambert, Steve
CD ROM: The New Papyrus
Microsoft Press, 1986

(Loye 1984) Loye, David
The Sphinx and the Rainbow
Bantam Books Inc, 1984

(Mach 1960) Mach, Ernst
Space and Geometry
Open Court Publishing Company, 1960

(Mark 1985) Mark, Earl
Architecture in Real Time: The Interdiciplinary Use of Film,
Video & Computer Graphics for Representing Architecture.
SMVIS Thesis, M.I.T. 1985

(Minsky 1986) Minsky, Marvin
The Society of Mind
Simon and Schuster, New York 1986

(Negroponte 1972) Negroponte, Nicholas
The Architecture Machine
M.I.T. Press, 1972

(Paivio 1971) Paivio, Allan
Imagery and Verbal Processes
Holt, Rinehart and Winston, Inc. 1971

(PC World 1987) PC World
CD ROM: Special Report
April, 1987

(PA 1984) Progressive Architecture
"Computers in Architecture"
May, 1984

(PA 1985) Progressive Architecture
"Computers in Architecture"
May, 1985

(Rand 1971) Rand, Ayn
The Romantic Manifesto
Bantam Books, 1971

(Rasmussen 1962) Rasmussen, Steen Eiler
Experiencing Architecture
M.I.T. Press, 1962

(Shannon 1963) Shannon, Claude E.
The Mathematical Theory of Communication
University of Illinois Press, 1963

(Simon 1969) Simon, Herbert A.
The Science of the Artificial
M.I.T. Press, 1969

(Tinney 1987) Tinney, Robert
"AT&T's TrueVision Image Processing System"
BYTE, March, 1987